PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

Título

DESARROLLO DE UN APLICATIVO MÓVIL PARA LA EXTRACCIÓN AUTOMÁTICA DE INFORMACIÓN DEL DOCUMENTO DE IDENTIFICACIÓN MEDIANTE VISIÓN COMPUTACIONAL

TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAGÍSTER EN INFORMÁTICA CON MENCIÓN EN CIENCIAS DE LA COMPUTACIÓN

AUTOR

Tomas Ivan Carrillo Fuertes

ASESOR

Dr. César Armando Beltrán Castañón

Octubre, 2019

RESUMEN

Los seres humanos han tenido varias formas de identificar individuos y grupos, pero los orígenes del sistema de identificación nacional contemporáneo se remontan a 1938, cuando los legisladores en el Reino Unido aprobaron la Ley de Registro Nacional, que exigía que todos los residentes posean tarjetas de identidad. Posteriormente, Alemania, Francia, Polonia, Grecia, entre otros, también instituyeron sistemas de tarjeta de identidad (ID) después del final de la Segunda Guerra Mundial. En Perú, el Documento Nacional de Identidad (DNI) constituye la única cédula de identidad personal reconocida por el Estado para todos los actos civiles, comerciales, administrativos, judiciales y, en general, para todos aquellos casos en que, por mandato legal, deba ser presentado. Su emisión se encuentra a cargo del Registro Nacional de Identificación y Estado Civil - RENIEC.

Así, la presentación del DNI implica sobre todo verificar la información que se encuentra en ella, para certificar que pertenece al portador. Para empresas como de seguros, bancos, clínicas, entre otros; es importante realizar esta tarea de verificación de forma rápida y precisa, pero por lo general esta tarea es realizada de forma manual o mediante el uso de la aplicación que facilita la RENIEC, pero la cual requiere una autorización además que posee un costo por consulta[1]. También existen diversos aplicativos OCR para leer información de las personas a partir de tarjetas de presentación, pasaporte y documentos de identidad, sin embargo, estas herramientas son tecnologías cerradas y propietarias, cuyos costos se hacen elevados en el tiempo. Por ello, el presente trabajo busca desarrollar un modelo algorítmico que permita extraer la información de una persona a partir de la imagen de su DNI electrónico.

Para conseguir este objetivo, el aplicativo utiliza algoritmos de procesamiento de imagen para identificar los datos de la persona en el DNI, luego separa cada dato de la persona en palabras y por último cada palabra en letras. Cada imagen que contiene una letra será clasificada por un modelo para identificar que letra es. En este trabajo

[1] https://cel.reniec.gob.pe/celweb/resources/img/Requisitos.pdf

para la clasificación de letras se probó los modelos de Adaboost, basado en árboles de decisiones y YOLO (v3 tiny), que es una arquitectura neuronal inspirada en GoogLeNet.

Tomando una muestra de 17 DNI electrónicos se obtuvo como resultado el 87% de letras correctamente detectadas con Adaboost y 98% con YOLO. En base a este resultado se concluye que los modelos Adaboost y YOLO pueden mejorar la extracción de la información de una persona a partir de la imagen de su DNI electrónico.

Palabras clave: documento de identidad, reconocimiento de caracteres, extracción de información, desarrollo de modelo OCR, Adaboost, YOLO.

ÍNDICE

Índice de Figuras

Índice de Tablas

Índice de Fórmulas

Índice de Códigos

INTRODUCCIÓN

Para las empresas recolectar información incorrecta de sus clientes no sólo representa elevados costos y pérdida de tiempo al gestionarlas, sino también pérdida de oportunidades para atraer, analizar y retener clientes. Por ejemplo, una empresa de seguros tiene un cliente importante: Juan, que es casado, pero en su sistema dice que es Julia y es soltera, cuando la empresa decida realizar un análisis de sus clientes más importantes, entre ellos Juan, el resultado de este análisis aportará poco valor debido a que, Según el estudio de Integrate (Integrate, 2019), así como con Juan hay un 40% Leads (clientes potenciales) afectados por un 25% de data errada en el sistema. Más aún, cuando se contacten para ofrecerle otro seguro o renovar el que ya cuenta llamándolo Julia, Juan pensará que no está dirigido a él y lo ignorará.

Para atender este tipo de necesidades existe una variedad de soluciones, entre ellas los aplicativos OCR que permiten extraer la información de una persona a partir de la imagen de su documento de identidad. Sin embargo, al usar estos aplicativos en un DNI Electrónico (DNIe) peruano no siempre se logra extraer la información de la persona de forma correcta, por ejemplo, al utilizar un aplicativo sobre el DNIe identifica el segundo nombre como apellido (Anexo ABBY OCR en DNI electrónico peruano). Por ello el presente trabajo busca mejorar el proceso de extracción de información de la persona a partir de la imagen de su DNI electrónico peruano (nombre completo, estado civil y género).

Al realizar una revisión sistémica de la literatura encontramos que hay varios estudios para extraer información personal como nombres y apellidos en base a su documento, y hay muchos trabajos sobre reconocimiento de caracteres donde se observa que podemos obtener una mayor precisión en los modelos cuya fuente son imágenes con menos variación de color, fuente, tamaño (Anjerome, et al., 2016) (Isaza, Vargas, Gaviria, & Hernández, 2012), se encontró también que el entrenamiento con caracteres de otro idioma pueden mejorar el rendimiento del modelo (J, Chen, Feng, & Xu, 2014). Los actuales métodos que se vienen usando y sus rendimientos para sus casos de aplicación son: usando SIFT y lógica difusa para detectar caracteres en

placas sobre un data set propio sé obtuvo 90.75% (Anjerome, et al., 2016), usando HOG-KNN para detectar caracteres en el data set ICDAR 2003 se obtuvo 80.4% (González, Bergasa, Yebes, & Bronte, 2012), usando Template Matching para detectar caracteres en un DNI Colombiano usando un data set propio se obtuvo 92.6% (Isaza, Vargas, Gaviria, & Hernández, 2012), usando ABBYY OCR para detectar caracteres en imágenes en el data set ICDAR 2011 se obtuvo 72.33% (Su, Lu, Phan, & Tan, 2012), usando SHL-CNN para detectar caracteres en diferentes idiomas en el data set ICDAR 2003 (J, Chen, Feng, & Xu, 2014), usando SVM-HOG para detectar caracteres en imágenes de SPAM en el data set Char74K se obtuvo 94.2% y en ICDAR 2003 86.01% (Naiemi, Ghods, & Khalesi, 2019), para detectar letras en el documento de identidad de Indonesia en una base de datos propia de 10000 imágenes de caracteres se obtuvo usando CNN 89% y SVM 71% (Pratama, Satyawan, Fajar, Fikri, & Hamzah, 2018), para detectar letras en una tarjeta de identidad china: en una base de datos propia de 363210 imágenes de caracteres se obtuvo usando LeNet 73%, CNN 90.4%, ResNet-56 99.7% (Xu & Wu, 2018), en una base de datos propia de 20 imágenes de documentos usando Template Matching y SVM se obtuvo 70% (Fang, Fu, & Xu, 2017), en una base de datos propia de 94125 imágenes de caracteres usando RSCNN se obtuvo 92.7% (Xiang, et al., 2018), para identificar letras en un documento de identidad de Kazajistán en una base de datos propias de 20770 imágenes de caracteres usando CCN se obtuvo 96.83% (Amirgaliyev, Kuatov, & Baibatyr, 2017), para identificar letras en un documento de Vietnam en una base de datos propia de 1000 imágenes de documentos usando RNN se obtuvo 85.7% (Liem, et al., 2019).

Además, se identificó los productos existentes de OCR: OmniPage de Nuance (Nuance, OmniPage Ultimate, 2019), Paperport de Nuance (Nuance, Paperport Professional 14, 2019), Abbyy FineReader 12 Professional (ABBYY, 2019), Soda PDF Business (Software, 2019), ID mobile de iCar (ICAR, 2019) y ABBYY Business Card Reader (ABBYY, 2019). Así como las soluciones en la nube de OCR: API Vision - Detección de texto en documentos (Google, 2019), Amazon Rekognition - Text in Image (Amazon, 2019), Computer Vision API v2.0 OCR (Microsoft, 2019) y Visual recognition - Text Model (IBM, 2019).

Para el presente proyecto se tendrán 3 etapas: la primera se detalla en el capítulo 3, utilizando técnicas de procesamiento de imágenes se identifica las regiones del DNI que contienen un dato de la persona, cada dato es previamente procesado para separar el fondo de las letras y se identifican sus palabras y letras. En el capítulo 4 se verá la segunda etapa, donde se entrenará los modelos para clasificar las letras: Adaboost (usando características de Haar) y YOLO v3 Tiny. Finalmente, en el capítulo 5 se detalla como cada letra de un dato de la persona será clasificada por los modelos: Adaboost usando sus características de Haar y YOLO en un entorno móvil.

Los resultados de estos 2 modelos son presentados en el capítulo 6, y el análisis de los resultados, así como las conclusiones y recomendaciones son comentados en el capítulo 7.

Para conseguir extraer la información de la persona a partir de la imagen de su DNI, el aplicativo utiliza algoritmos de procesamiento de imagen para identificar los datos de la persona en el DNI, luego separa cada dato de la persona en palabras y por último cada palabra en letras. Cada imagen que contiene una letra será clasificada por un modelo para identificar que letra es. En este trabajo, para la clasificación de letras se probó los modelos de Adaboost, basado en árboles de decisiones (Viola & Jones, 2001) y YOLO (v3 tiny), que es una arquitectura neuronal inspirada en GoogleLeNet (Redmon, Divvala, Girshick, & Farhadi, 2016).

Tomando una muestra de 17 DNI electrónicos se obtuvo como resultado el 80% de datos correctamente detectados con Adaboost y 98% con YOLO, en comparación a otros métodos que tienen un 75% de acierto al recolectar información, según el estudio de Integrate (Integrate, 2019). En base a este resultado se concluye que los modelos Adaboost y YOLO pueden mejorar la extracción de la información de una persona a partir de la imagen de su DNI electrónico.

CAPÍTULO 1

1 Generalidades

1.1 Problemática

Actualmente en Perú, las empresas tienen graves problemas con respecto a la información de sus clientes, la cual por lo general es incorrecta o no está actualizada, esto genera clientes insatisfechos y gastos por solucionar problemas por información incorrecta.

La RENIEC brinda servicios para obtener esta información; sin embargo, es una solución costosa ($0.70 por consulta) y no es completa ni permite consultar DNI de menores de edad (excepto MINSA). Las empresas grandes vienen invirtiendo en mejorar la calidad de su data, lo cual les genera un costo anual de $ 2 millones aproximadamente (consultado a una empresa del sector de seguros de forma anónima) o recurren a comprar data de las personas (ejemplo Equifax), sin embargo, no tienen todos los datos del DNI. Se pueden identificar los siguientes efectos del problema:

- Extracción incorrecta de los datos de la persona
- Extracción incompleta de los datos de la persona
- Costos por corrección de datos de las personas
- Demora en procesos por corrección de datos de las personas

Estos efectos son la manifestación del problema, que se puede enunciar como: “No existe un aplicativo que permita extraer la información de una persona a partir de la imagen de su DNI electrónico de forma automática, rápida y en tiempo real”

Las principales causas del problema son:

1. No existe un modelo que identifique los datos de un DNIe dentro de una imagen. Cada tipo de documento tiene diferentes datos de una persona y una

distribución específica, debido a esto algunos aplicativos genéricos a veces identifican correctamente el valor de un dato personal, pero lo asignan a otro dato la persona.

2. No existe un modelo que segmente las palabras y letras de cada dato del DNIe. Una imagen del DNIe peruano presenta mucho ruido debido a la imagen de fondo y al desenfoque propio de las cámaras con las que es captada, por lo que es ambiguo si el dato está compuesto por 1 o más palabras, y la cantidad de letras de cada palabra. Sin esta detección correcta el segundo nombre de una persona podría ser omitido o confundido como otro dato.

3. No existe un modelo que clasifique una imagen dentro de un DNIe en una letra. Para clasificar imágenes en letras existen modelos complejos que requieren mucho computo lo cual no siempre es adaptable a un entorno móvil, en otros casos el tiempo que demoran en realizar la clasificación en un entorno móvil es excesivo para ser una solución viable.

4. No existe un aplicativo que orqueste el modelo que identifica el dato, detecta las palabras y letras de cada dato, y que reconoce las letras del dato de un DNIe. Muchos modelos de aprendizaje maquinan no son fáciles de portar desde un entorno de computadora a un entorno móvil o son cajas negras que no tienen un óptimo rendimiento en un DNIe peruano.

1.1.1 Objetivo general

Desarrollar un prototipo computacional móvil que permita extraer los datos de una persona a partir de la imagen de su DNI electrónico peruano, usando modelos de aprendizaje máquina para la detección y clasificación de letras.

1.1.2 Objetivos específicos

Los objetivos específicos son:

- **(O1)** Desarrollar un modelo que segmente las palabras y letras en una imagen de DNI electrónico dada.
- **(O2)** Diseñar e implementar los modelos predictivos para clasificar una imagen en una letra.
- **(O3)** Diseñar e implementar un prototipo móvil que lea el DNI electrónico, segmente las letras usando el modelo de (O1), clasifique cada letra usando el modelo predictivo de (O2) y presente los resultados.

1.1.3 Resultados esperados

Los resultados esperados y los medios de verificación para cada uno son:

Tabla 1.1 Objetivos, sus resultados esperados y medios de verificación

	OBJETIVO GENERAL	RESULTADOS FINALES	MEDIOS DE VERIFICACIÓN
0	Desarrollar un prototipo computacional móvil que permita extraer los datos de una persona a partir de la imagen de su DNI electrónico peruano, usando modelos de aprendizaje máquina para la detección y clasificación de letras.	Aplicativo móvil desarrollado que permite extraer los datos de una persona a partir de la imagen de su DNI electrónico.	**Indicador 5**: datos correctamente identificados.
	OBJETIVOS ESPECÍFICOS	**RESULTADOS INTERMEDIOS**	**MEDIOS DE VERIFICACIÓN**
1	Desarrollar un modelo que segmente las palabras y letras en una imagen de DNI electrónico dada.	Modelo desarrollado que segmenta las palabras y letras de cada dato de la persona.	**Indicador 1**: número de palabras en cada dato segmentados correctamente. **Indicador 2**: número de letras en cada palabra segmentados correctamente.
2	Diseñar e implementar los modelos predictivos para clasificar una imagen en una letra.	Modelos YOLO y Adaboost desarrollados que clasifican una imagen en letra.	**Indicador 3**: letras clasificadas correctamente.
3	Diseñar e implementar un prototipo móvil que lea el DNI electrónico, segmente las letras usando el modelo de (O1), clasifique cada letra usando el modelo predictivo de (O2) y presente los resultados.	Aplicativo móvil desarrollado que lee un DNI electrónico, segmenta, clasifica las letras de cada dato personal y presenta el resultado.	**Indicador 4**. Letras correctamente clasificadas y posicionadas en el dato de la persona.

1.2 Herramientas, métodos y metodologías

1.2.1 Introducción

Las herramientas, métodos y modelos usados en el proyecto para alcanzar cada resultado intermedio son los siguientes:

Tabla 1.2 Mapeo de Resultados vs. Herramientas a utilizarse.

Resultado intermedio	Herramientas
(RE1) Modelo desarrollado que detecta las palabras y letras de cada dato de la persona.	-Anaconda Navigator (Jupyter) -Python -OpenCV (Python)
(RE2) Modelos YOLO y Adaboost desarrollados que detectan y clasifican una imagen en letra.	-Anaconda Navigator (Jupyter) -Python -Modelo Adaboost -Método Imagen Integral -Método descriptor de Haar -Modelo de Adaboost -Modelo YOLO v3 tiny -Validación cruzada -OpenCV (Python)
(RE3) Aplicativo móvil desarrollado que lee un DNI electrónico, detecta, clasifica las letras de cada dato personal y presenta el resultado.	- Android Studio - Java - OpenCV para Android

1.2.2 Herramientas

1.2.2.1 Python

Python es un lenguaje de programación interpretado multiplataforma, multiparadigma (programación orientado a objetos, programación imperativa y programación

funcional) es administrado por Python Software Foundation y es de código abierto bajo la licencia de Python Software Foundation Licence y es compatible con la licencia pública de GNU a partir de la versión 2.1.1.

1.2.2.1 Anaconda Navigator

Anaconda es una plataforma de desarrollo en Python enfocado en procesamiento de data a larga escala, análisis de datos y minería de datos; así como en la gestión de librerías Python.

Su selección se basa en que presenta librerías que nos serán útiles para el desarrollo del siguiente trabajo en especial con los paquetes:

- **PIL (Python Image Library)** librería de Python enfocada al tratamiento de imágenes.
- **scikit-image** librería que alberga algoritmos para el procesamiento de imágenes.
- **scikit-learn** librería que alberga algoritmos para análisis de datos y minería de datos.

“Anaconda Navigator” es un gestor de entornos y librerías de Python, adicionalmente cuenta con herramientas muy útiles ya integradas como “Jupyter” el cual permite editar y ejecutar código Python de forma rápida y fácil.

1.2.2.2 OpenCV

OpenCV es un conjunto de librerías para desarrollo de modelos de aprendizaje máquina y tratamiento de imágenes. Los beneficios de OpenCV es que nos permite entrenar los modelos en una computadora o en la nube usando código C, Python o Java. Una vez finalizado el entrenamiento del modelo en la computadora o en la nube (servicio de alquiler de capacidad de computo), este puede ser guardado en un formato XML, y ser reutilizados en otro entorno como un aplicativo móvil con las librerías OpenCV iOS u OpenCV Android.

1.2.3 Métodos

1.2.3.1 Método descriptor de Haar

El descriptor de Haar se obtiene al aplicar en una imagen una serie de filtros en diferentes escalas para obtener las características de la imagen a usar en el modelo.

Los filtros a usar presentan las siguientes características:

- Conjunto de rectángulos del mismo tamaño contiguos horizontal y/o verticalmente.
- Los rectángulos negros representan zonas con una contribución negativa al valor filtro.
- Los rectángulos blancos representan zonas con una contribución positiva al valor filtro.
- El resultado del filtro es la diferencia de los valores de los pixeles entre las zonas de negro y las zonas de blanco.

Un ejemplo de los filtros básicos de Haar se muestran en la Figura 1.1.

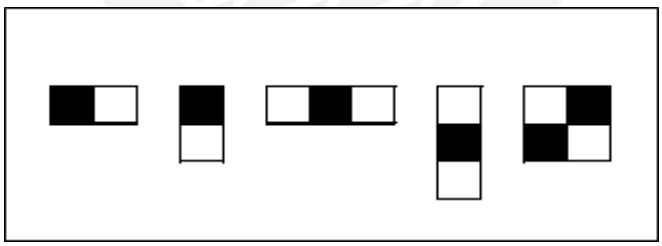


Figura 1.1 Los 5 filtros básicos de Haar, son mascarás que se superpondrán a los pixeles de la imagen, donde los pixeles de la parte oscura tendrán una contribución negativa al valor del filtro y de la parte blanca una contribución positiva.

Ejemplo de cálculo de un filtro de Haar en una imagen 2 x 4.

Como se muestra en la

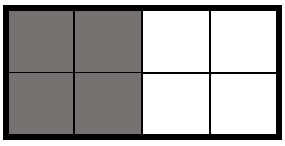


Figura 1.2, se tiene una imagen de 2 x 4, la intensidad de cada pixel está en medio de cada recuadro.

100	110	10	12
90	200	11	15

Figura 1.2 Arriba una Imagen de 2x4 pixeles con la intensidad de cada pixel y abajo un filtro de Haar de 2x4 con recuadros oscuros y blancos.

Al superponer el filtro de Haar sobre la imagen obtenemos el siguiente resultado:

100	110	10	12
90	200	11	15

Figura 1.3 Imagen con el filtro de Haar superpuesto.

Para calcular el valor del filtro se sumarán la intensidad de los pixeles en la zona blanca y la intensidad de los pixeles en la zona oscura y luego se procederá a restar ambas sumas.

valor del filtro = $\sum_{(x,y)\in N} I(x,y) - \sum_{(x,y)\in B} I(x,y)$

valor del filtro = (100 + 110 + 90 + 200) - (10 + 12 + 11 + 15)

valor del filtro = 452

Para interpretar los filtros de Haar y las características que revelan de una imagen podemos ver en la Figura 1.4 el resultado de su aplicación sobre la imagen de un rostro. El primer filtro horizontal muestra los contornos horizontales siendo más notorio la nariz, el filtro vertical muestra los contornos verticales del rostro haciendo más notorio las cejas y la boca, un filtro vertical más ancho tenderán a ensanchar también los contornos verticales. Por otro lado, un filtro de horizontal de tres mostrará más contornos horizontales, un filtro vertical de tres también mostrará más contornos verticales, por último, un filtro diagonal mostrará los contornos diagonales en el rostro.

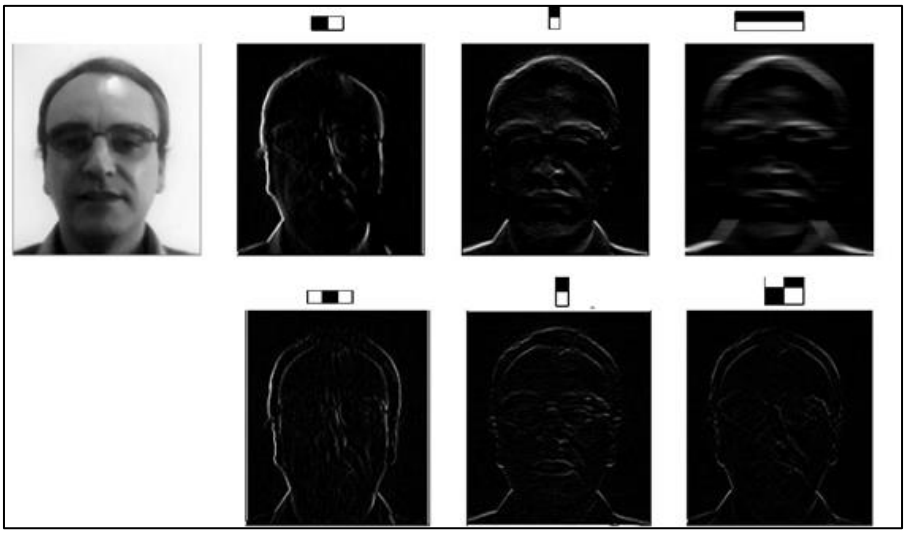


Figura 1.4 Aplicación de los filtros básicos de Haar a una imagen en escala de grises. Extraído de (López, Valveny, & M, 2019)

Los filtros de Haar dada la forma como estos se calculan permiten detectar contrastes en la dirección horizontal, vertical, en diagonal y a diferentes rangos y escalas.

Aplicación de los filtros de Haar a una imagen:

- Cada filtro de Haar se aplica a todas las posibles escalas en horizontal y vertical. Un ejemplo de las escalas se puede ver en la Figura 1.5.
- Cada filtro escalado se aplica en todas las posibles posiciones de la imagen.
- El resultado de aplicar cada filtro en cada escala y posición posible es una característica de Haar.
- En imágenes de 24 x 24 píxeles se obtienen 162336 características.

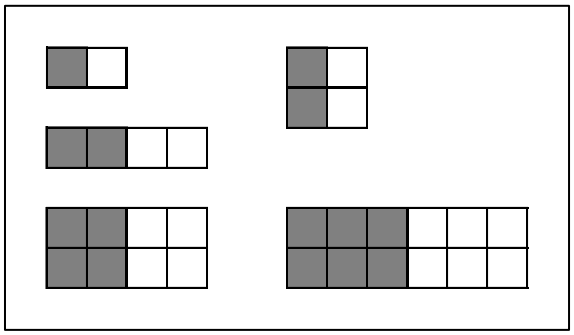


Figura 1.5 Filtro de Haar en diferentes escalas en horizontal y vertical.

Filtros de Haar extendidos

Son rotaciones de los filtros básicos de Haar que permiten detectar características direccionales. En la Figura 1.6 se puede apreciar los filtros básicos de Haar y los extendidos y como estos últimos son una rotación de los filtros básicos.

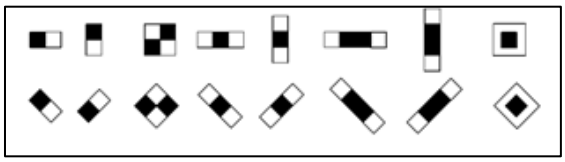


Figura 1.6 Filtros de Haar extendidos. Extraído de (López, Valveny, & M, 2019)

1.2.3.2 Imagen Integral

El método de imagen integral consiste en colocar en cada pixel de la imagen integral la suma de todos los píxeles situados a la izquierda y arriba en la imagen original. Este método permite calcular de forma muy rápida la suma de pixeles que conforman un rectángulo de cualquier dimensión en la imagen original. En la Figura 1.7 el valor del pixel en la posición (x,y) estará dada por la suma de todos los pixeles en la zona sombreada, lo cual se puede expresar por la Fórmula 1.1.

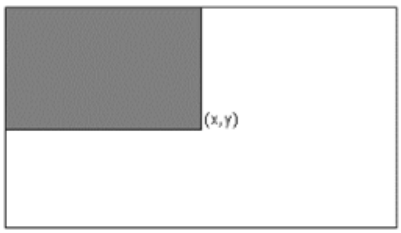


Figura 1.7 Representación gráfica de una imagen integral.

$$II(x,y) = \sum_{x' \le x, y' \le y} I(x',y')$$

Fórmula 1.1 Cálculo de un pixel de la imagen integral (II) en base a los píxeles de la imagen original (I).

Cálculo de la imagen integral

El cálculo es muy eficiente pues requiere un sólo recorrido por toda la imagen, para ello hacemos uso de la suma acumulada de la fila actual y el valor de la imagen integral del pixel arriba del pixel a calcular. Por ejemplo, en la Figura 1.8 el valor del pixel rojo (x,y) en la imagen integral es la suma de dos zonas: la zona verde más la suma de la

zona celeste incluyendo el cuadrado rojo.

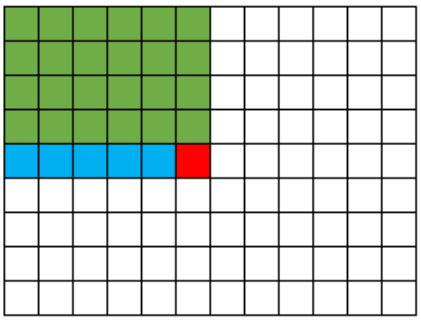


Figura 1.8 Ejemplificación del cálculo de la imagen integral como la suma de la imagen integral superior y la suma acumulada.

La suma de la zona verde se puede expresar como la Imagen Integral del pixel superior II(x,y-1) y la suma de la zona celeste se puede expresar como la suma de los pixeles a la izquierda s(x-1,y) más el pixel actual I(x,y) como se muestra en la Fórmula 1.2.

$$II(x,y) = II(x,y-1) + S(x,y)$$

$$S(x,y) = \sum_{x' \leq x} I(x',y) = S(x-1,y) + I(x,y)$$

Fórmula 1.2 Fórmula de cálculo de la Imagen integral en un solo recorrido.

Utilización de la imagen integral

Sea “A” la suma de pixeles de un recuadro cualquiera en la imagen cuyas esquinas son los puntos: P1, P2, P3 y P4, como se muestra en la Figura 1.9, su cálculo estará dado por la Fórmula 1.3 en donde será igual a 3 operaciones aritméticas por lo que siempre será un tiempo constante.

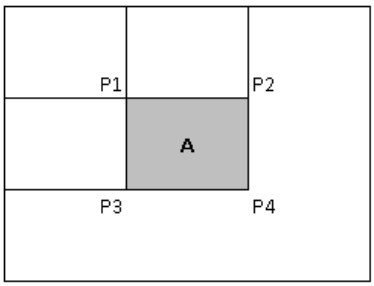


Figura 1.9 Cálculo de suma de intensidad de pixeles usando Imagen Integral.

$$A = II(P_4) - II(P_2) - II(P_3) + II(P_1)$$

Fórmula 1.3 Fórmula para calcular la suma de pixeles en un rectángulo de la imagen

1.2.3.3 Adaboost

El modelo de Adaptive boosting (Adaboost) combina clasificadores simples (un poco mejores que una clasificación aleatoria). Cada clasificador simple o también conocido como clasificador débil se aprende dando un peso diferente a cada muestra. (Viola & Jones, 2001)

Para el ensamble del modelo fuerte se irán generando clasificadores débiles siguiendo los siguientes pasos:

- Se da un mayor peso a los casos mal clasificados por los clasificadores previos.
- Se da un menor peso a los ejemplos bien clasificados por los clasificadores previos.

Como se muestra en la Figura 1.10 se tienen dos clases, diferenciados por los colores, siguiendo los pasos sería:

1. Un primer clasificador débil predice correctamente los azules al lado izquierdo y mal algunos azules del derecho.
2. Para la segunda iteración se da más peso a los azules mal clasificados (mayor tamaño) y se da menos peso a los azules que fueron bien clasificados (menor tamaño), tomando en cuenta los nuevos pesos el segundo clasificador identifica correctamente los rojos de la derecha y mal los rojos de la izquierda.
3. Para la tercera iteración se da más peso a los rojos mal clasificados y menos peso a los rojos bien clasificados y el tercer modelo separa verticalmente los puntos prediciendo rojos en la parte inferior y azules en la parte superior.
4. Luego se procede a ensamblar estos tres clasificadores mejorando la predicción global.

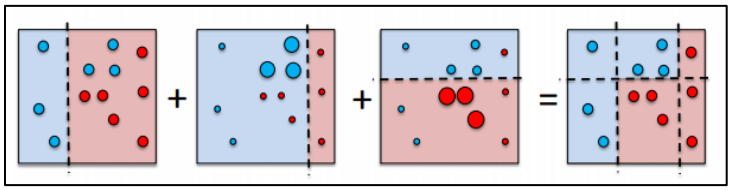


Figura 1.10 Ejemplo de 3 clasificadores débiles que combinados conforman un clasificador global fuerte con mejor rendimiento, se aprecia el cambio de los pesos en relación al tamaño en cada iteración. Extraído de (López, Valveny, & M, 2019)

1.2.3.4 YOLO

El modelo de Yolo es una red neuronal profunda que fue inspirada de GoogLeNet cuenta con 24 capas convolucionales y 2 full conectadas como se puede apreciar en Figura 1.11. Las primeras capas convolucionales ayudan a extraer las características de la imagen mientras que las capas full conectadas se encargan de generar las probabilidades por clase y las coordenadas de la caja (Redmon, Divvala, Girshick, & Farhadi, 2016).

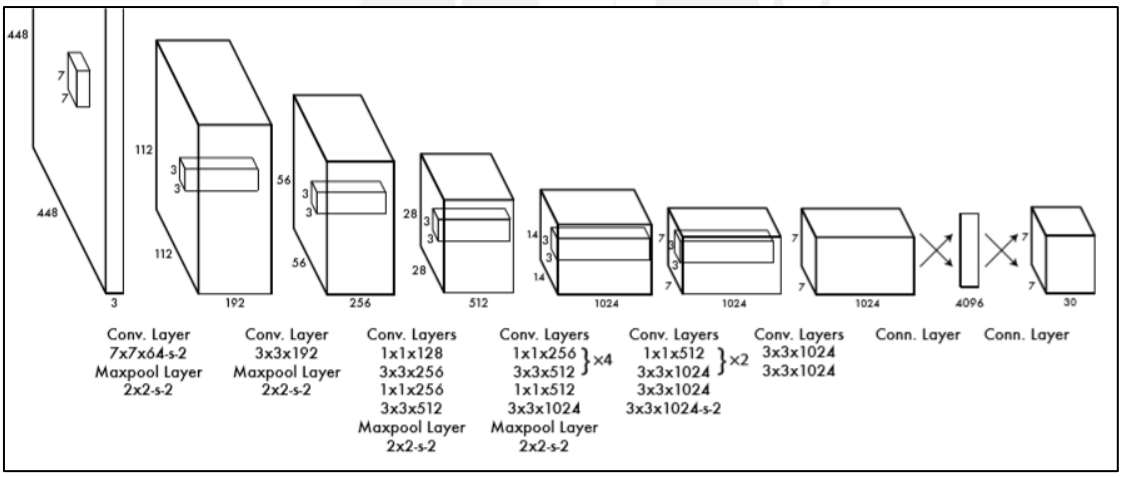


Figura 1.11 Arquitectura de YOLO con 24 capas convoluciones y 2 full conectadas. Extraído de (Redmon, Divvala, Girshick, & Farhadi, 2016)

Actualmente YOLO cuenta con 3 versiones una arquitectura alterna más ligera “tiny”, las versiones son:

- **YOLO v1** que cuenta con una arquitectura de 24 capas convolucionales y 2 full conectadas.
- **YOLO v2** se adapta a la estructura de Darknet, una red neuronal de código abierto (Redmon & Farhadi, Darknet: Open Source Neural Networks in C,

2019), y se remueve las 2 capas full conectadas convirtiéndolo en una red totalmente convolucional, mejorando su diseño para imágenes de mayor resolución y el rendimiento en la detección respecto a YOLO v1, de 63.4 a 78.6 mAP sobre la data PASCAL VOC 2007 (Pascal, 2019) según (Redmon & Farhadi, YOLO9000: Better, Faster, Stronger, 2017).

- **YOLO v3** cambia su función de perdida de “classification loss” por “binary cross-entropy loss” con lo cual reduce el tiempo de computo en entrenamiento y cambia su arquitectura por una estructura con 53 capas convolucionales llamado Darknet-53 y **YOLO v3 tiny** es una adaptación de YOLO v3 donde se reduce las capas a 23, disminuyendo la precisión y aumentando la velocidad de detección. (Redmon & Farhadi, YOLO: Real-Time Object Detection, 2019).

El modelo (YOLO v3) es aproximadamente 1000 veces más rápido que un R-CNN y 100 veces más rápido que Fast R-CNN (Redmon & Farhadi, YOLO: Real-Time Object Detection, 2019).

Para realizar la detección enfoca el problema como una regresión, primero divide la imagen en una grilla de tamaño SxS, cada celda de la grilla se encarga de realizar la predicción de B cajas envolventes y sus probabilidades; cada caja envolvente cuenta con 5 predicciones: x,y,w,h y la probabilidad; x e y son las coordenadas relativas del centro de la caja respecto a la imagen, w y h son el ancho y largo relativos de la caja respecto a la imagen. Luego por cada celda de la grilla se calcula C probabilidades por cada clase dado que hay un objeto en la celda, lo que nos da un mapa de probabilidades por cada clase en cada celda de la grilla (S, S, B×5 + C) y en base a estas se obtiene las detecciones finales como se muestra en la Figura 1.12.

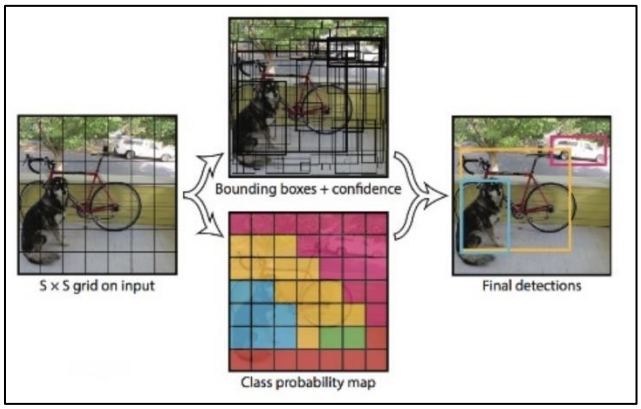


Figura 1.12 Para la detección primero Imagen es dividida en SxS celdas, se calculan B cajas envolventes a cada celda, por cada celda se calculan C probabilidades de que sea una clase dato que hay un objeto en la celda y finalmente se obtienen las detecciones . Extraído de (Redmon & Farhadi, YOLO: Real-Time Object Detection, 2019)

Tabla 1.3 Arquitectura de Yolo v3 Tiny

layer	filters	size	input	output
0 conv	16	3 x 3 / 1	96 x 96 x 3 ->	96 x 96 x 16 0.008 BF
1 max		2 x 2 / 2	96 x 96 x 16 ->	48 x 48 x 16 0.000 BF
2 conv	32	3 x 3 / 1	48 x 48 x 16 ->	48 x 48 x 32 0.021 BF
3 max		2 x 2 / 2	48 x 48 x 32 ->	24 x 24 x 32 0.000 BF
4 conv	64	3 x 3 / 1	24 x 24 x 32 ->	24 x 24 x 64 0.021 BF
5 max		2 x 2 / 2	24 x 24 x 64 ->	12 x 12 x 64 0.000 BF
6 conv	128	3 x 3 / 1	12 x 12 x 64 ->	12 x 12 x 128 0.021 BF
7 max		2 x 2 / 2	12 x 12 x 128 ->	6 x 6 x 128 0.000 BF
8 conv	256	3 x 3 / 1	6 x 6 x 128 ->	6 x 6 x 256 0.021 BF
9 max		2 x 2 / 2	6 x 6 x 256 ->	3 x 3 x 256 0.000 BF
10 conv	512	3 x 3 / 1	3 x 3 x 256 ->	3 x 3 x 512 0.021 BF
11 max		2 x 2 / 1	3 x 3 x 512 ->	3 x 3 x 512 0.000 BF
12 conv	1024	3 x 3 / 1	3 x 3 x 512 ->	3 x 3 x1024 0.085 BF
13 conv	256	1 x 1 / 1	3 x 3 x1024 ->	3 x 3 x 256 0.005 BF
14 conv	512	3 x 3 / 1	3 x 3 x 256 ->	3 x 3 x 512 0.021 BF
15 conv	93	1 x 1 / 1	3 x 3 x 512 ->	3 x 3 x 93 0.001 BF
16 yolo				
17 route	13			
18 conv	128	1 x 1 / 1	3 x 3 x 256 ->	3 x 3 x 128 0.001 BF
19 upsample		2x	3 x 3 x 128 ->	6 x 6 x 128
20 route	19 8			
21 conv	256	3 x 3 / 1	6 x 6 x 384 ->	6 x 6 x 256 0.064 BF
22 conv	93	1 x 1 / 1	6 x 6 x 256 ->	6 x 6 x 93 0.002 BF
23 yolo				

1.2.3.5 Validación cruzada

La validación cruzada o *cross-validation* consiste en dividir la muestra en n bloques de igual tamaño y usar n-1 bloques para entrenar el modelo y el bloque no usado para el entrenamiento es usado para la validación. Este proceso es iterativo usando todos los bloques para el *testing*, esta técnica nos permite poder evaluar diferentes modelos o ajustar hiperparámetros al entrenar el modelo.

1.2.4 Metodología y plan de trabajo

Para el presente trabajo de tesis se tiene la siguiente estructura de trabajo:

1.2.4.1 Estudio de la literatura

En esta primera etapa del proyecto obtenemos los conocimientos requeridos para el desarrollo el presente trabajo. Para ello se hizo la búsqueda de los términos relacionados a la computación gráfica, el estado del arte de las investigaciones relacionadas al reconocimiento de caracteres, así como técnicas y herramientas usadas para atender la problemática planteada en este trabajo.

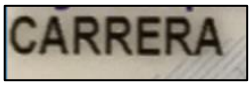
1.2.4.2 Adquisición de imágenes de DNI electrónicos

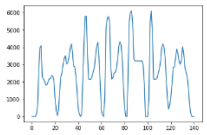
En esta etapa seleccionamos las imágenes a ser usadas para el entrenamiento de los modelos Adaboost y YOLO; y su validación de cada uno.

1.2.4.3 Desarrollo del modelo de extracción de datos personales

La etapa principal del trabajo es el desarrollo del modelo que permitirá obtener la información de un documento de identidad. Las etapas del aplicativo para extraer y presentar la información de la persona se detallan en la Tabla 1.4.

Tabla 1.4 Etapas del aplicativo para extraer y presentar la información de la persona.

Etapa	**Descripción**	**Ejemplo**
1. Adquisición	Lectura de la imagen del DNI	
2. Localización	Localizar solo la región del dato	
3. Preprocesamiento	Filtro de intensidad de color	
	Eliminar ruido	

4. Detección	Detección de espacios entre las letras y palabras del dato.	
5. Extracción	Extraer la letra del dato	
6. Clasificación	Clasificar imagen en letra	Letra “O”
7. Presentación	Presentar los datos	Segundo Apellido: Carrera

1.2.4.4 Experimentación para la optimización del modelo

Con la finalidad de mejorar los resultados obtenidos por el modelo se realizarán una serie de experimentos en la extracción de los datos y entrenamiento del modelo, y con ello obtener un mejor desempeño del modelo al clasificar las letras y obtener así la información de la persona.

1.2.4.5 Evaluación del prototipo y conclusiones obtenidas

Al tener el prototipo se procederá a medir el rendimiento de cada componente y su propio rendimiento según los indicadores descritos en la tabla 1.1. Una vez presentados los resultados se emitirán las conclusiones y recomendaciones.

1.2.4.6 Documentación del trabajo realizado

A la par del desarrollo de este trabajo se documentará lo realizado dando la respectiva descripción de los pasos seguidos para que puedan ser entendidos y reproducibles. Se analizará los resultados obtenidos brindando y describiendo las conclusiones y se brindará recomendaciones de cómo el presente trabajo pudiera ser extendido.

1.3 Delimitación

1.3.1 Alcance

El presente trabajo tiene como objetivo el diseño, implementación, evaluación y optimización de un prototipo que permita extraer datos personales a partir de la imagen de un DNI electrónico. Para ello se tiene el siguiente alcance:

- El trabajo usará dos modelos Adaboost con el descriptor Haar y YOLO v3 tiny.
- El tipo de imagen a usar para detectar caracteres es el DNI electrónico de una persona adulta y solo se tomarán en cuenta las letras: “ABCDEFGHIJKLMNOPQRSTUVWXYZ” y no la “Ñ”.
- Las regiones a analizar serán: prenombres, primer apellido, segundo apellido, sexo y estado civil.
- Como el prototipo a construir debe ser un aplicativo móvil multiplataforma, deberá poder detectar un dato en menos de 3 segundos y trabajará sin conexión a internet. Para tal fin se usará la librería OpenCV que permite correr los modelos de Adaboost y YOLO dentro del mismo celular y es multiplataforma (Android y iOS).

CAPÍTULO 2

2 Marco conceptual

2.1 Definiciones

2.1.1 Documento Nacional de Identidad Electrónico (DNIe)

El Documento Nacional de Identidad electrónico (DNIe) es un documento nacional de identidad emitido por el Registro Nacional de Identificación y Estado Civil - RENIEC que acredita en forma física y electrónica la identidad de la persona dueña del documento.

El DNIe se diferencia del DNI en que permite la firma digital de documentos electrónicos, el ejercicio del voto electrónico presencial y no presencial en los procesos electorales, así como la verificación biométrica dactilar.

Los componentes de seguridad el DNI electrónico son: impresión irisada, micro línea offset, fondo anticopia, fondo numismático, guilloche, chip criptográfico, tinta ópticamente visible, CLI, laser engraving, DOVID, Microtexto offset con error deliberado, tal como se muestra en la Figura 2.1.

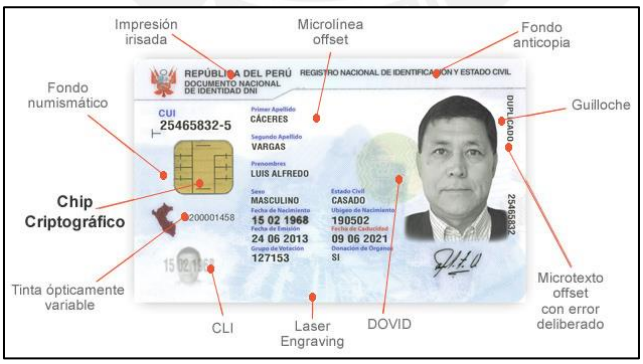


Figura 2.1 DNIe y 10 componentes de seguridad. Extraído de (RENIEC, 2019)

Un DNIe contiene zonas de información donde se encuentra la información de la persona que son relevantes para el presente trabajo como: Primer Apellido, Segundo Apellido, Prenombres, Sexo, Estado Civil, en adición a otros campos. En la Figura 2.2 se muestras estás zonas de información y otras adicionales.

Figura 2.2 DNIe con las zonas de información del DNI. Modificación de la imagen extraída de (RENIEC, 2019)

2.1.2 Imagen

Una imagen, para el presente trabajo, es una función f(x,y) conformada por las coordenadas espaciales: “x” e “y”, y la intensidad: f(x,y). Las coordenadas espaciales y la intensidad toman valores discretos (imagen digital). El elemento básico, un punto, es representado por sus coordenadas espaciales y su intensidad, también conocido como pixel. En Figura 2.3 se puede ver la representación de una imagen en pixeles.

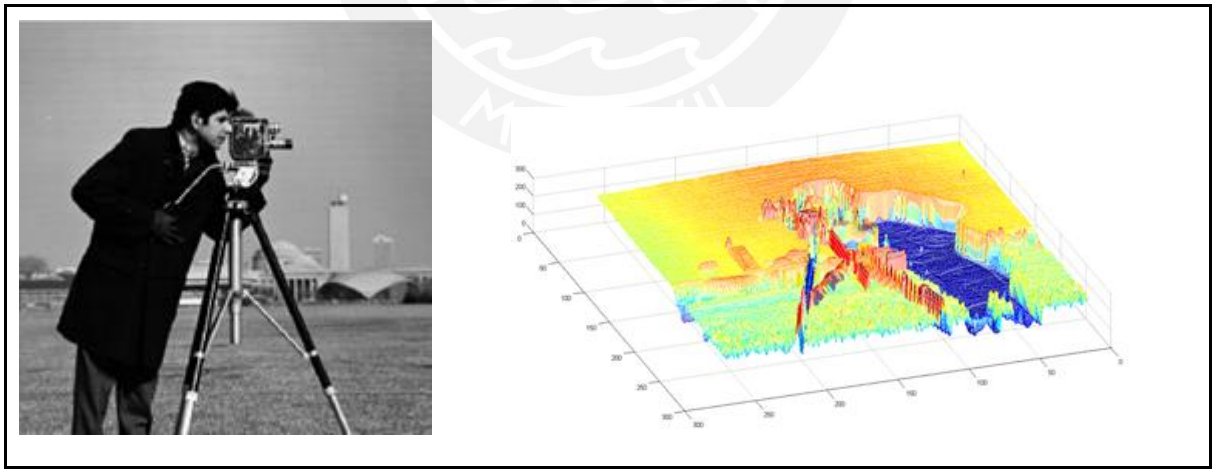


Figura 2.3 A la izquierda, la imagen de una persona, en la derecha una representación de los píxeles de la imagen con sus intensidades en el eje vertical. Extraído de (Gonzalez & Woods, 2008)

Muchas veces las imágenes son afectadas por 3 factores, la fuente de luz, las propiedades de reflexión de luz del objeto y la sensibilidad de receptor de luz por lo general una cámara con 3 canales de colores: rojo, verde y azul (RGB).

Así por ejemplo si:

- Longitud de onda en un instante dado: d($\boldsymbol{\lambda}$)
- Luz emitida: I($\boldsymbol{\lambda}$)
- % luz reflejada: S($\boldsymbol{\lambda}$)
- La sensibilidad de la cámara en cada canal es: R($\boldsymbol{\lambda}$), G($\boldsymbol{\lambda}$), B($\boldsymbol{\lambda}$)

Entonces la captación de luz final de la cámara será:

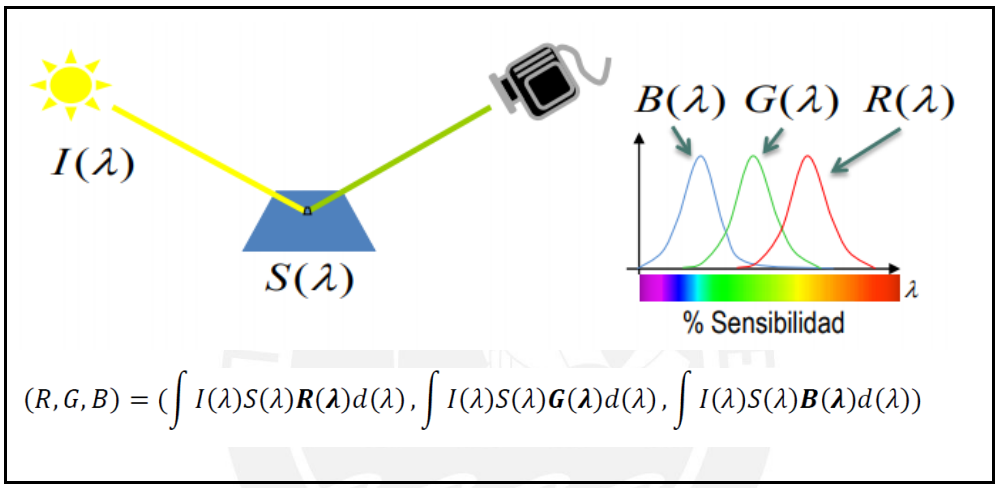


Figura 2.4 Ecuación de la imagen percibida por la cámara en función de la intensidad de la fuente de luz "I", la sensibilidad a la luz de una superficie "S" y su sensibilidad a los canales de color RGB. Extraído de (López, Valveny, & M, 2019)

2.1.3 Visión por computadora

Los seres humanos percibimos con facilidad los objetos en nuestro entorno tridimensional y sus características como su iluminación, bordes, colores, sus tonalidades, translucidez. En una foto grupal uno puede contar las personas presentes en la imagen, identificarlos con sus nombres, incluso inferir sus estados de ánimos por sus expresiones corporales y sus gestos faciales.

La visión por computadora busca representar nuestra realidad a través de modelos matemáticos. Estos modelos pueden describirnos los colores, iluminación, transparencia de los objetos, etc. Por ejemplo, dado un haz de luz que toca una superficie y esta es reflejada en varias direcciones, esto se puede explicar con la

función de distribución de reflectancia bidireccional (BRDF por sus siglas en ingles), a través de modelos físicos (Torreance & Sparrow, 1967; Cook & Torrance, 1982; Glassner, 1995).

2.1.3.1 Detección de objetos

La detección de objetos busca determinar la existencia y localización un objeto de interés dentro de una imagen, con grado de confianza; por lo general siguiendo dos pasos: primero identifica aquellas características de la imagen que son las más relevantes para la detección, una vez logrado este paso, en base a estas características procede a determinar si el objeto de interés está presente en la imagen, así como su posición y tamaño.

¿Cómo se detectan los objetos?

- Usando las técnicas de visión por computador y
- Apariencia visual.

Los retos actuales de la detección automática de objetos son:

- Poder discriminar unas clases de otras (ejemplo: persona, caballo, perro, carro, gato, fondo)
- Tener robustez ante la variabilidad propia de la misma clase de un objeto.
- Tener robustez ante la variabilidad del entorno donde se busca la imagen

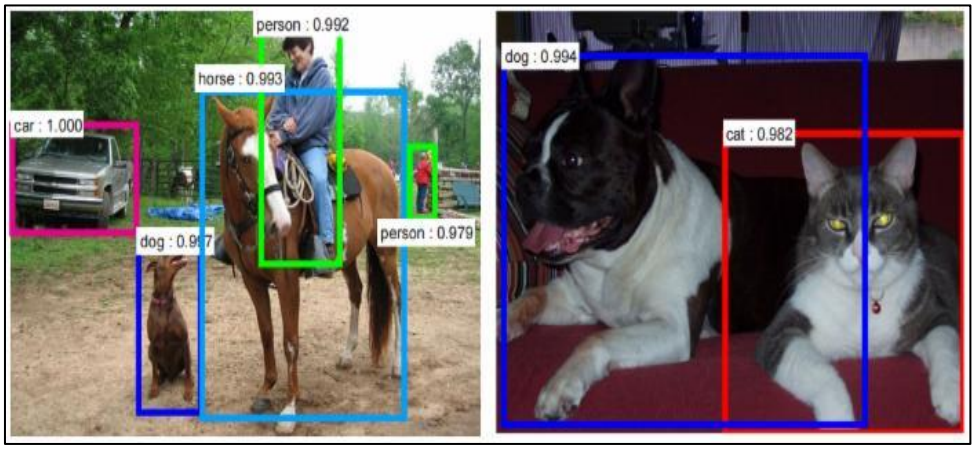


Figura 2.5 Ejemplo de detección de objetos en una imagen. Extraído de (Sinhal, 2019)

2.1.3.2 Categorización y etiquetado de imágenes

La categorización y etiquetado de imágenes busca determinar el contexto que describa mejor las escenas dentro de la imagen. Para lograrlo primero identifica que objetos están presentes, luego calcula la distribución espacial de todos los objetos dentro de la imagen, y por último en base a los objetos presentes y su distribución determina el contexto. Esté contexto estará representado por una etiqueta que se asignará a la imagen. Una de las cosas complejas en la categorización de imágenes es que los humanos podemos asignar varias categorías a una misma imagen

Figura 2.6 Imagen que puede ser etiquetada con: persona, bicicleta, puente, ciudad, agua, cielo, o incluso con etiquetas como: salto, ciclismo, peligro. Extraído de (Pascal, 2019)

¿Cómo se asignan etiquetas a una imagen?

- Proceso de aprendizaje a partir de imágenes de ejemplo.
- Aprender qué características son comunes a las imágenes de una categoría y las diferencias de las imágenes de otra categoría.

Los retos actuales de la clasificación de imágenes son:

- Tener robustez ante la variabilidad del contenido visual de imágenes: iluminación, tamaño, posición, punto de vista, fondo de la imagen, oclusión parcial, variabilidad intra clase.
- Número muy elevado de categorías.

2.1.3.3 Aplicaciones

Entre las principales aplicaciones tenemos:

- **Análisis de materiales microscópicos**

  Se usa fotos microscópicas de materiales para detectar determinados patrones que se pueden interpretar como fallas superficiales en un producto. Ejemplos de imágenes microscópicas para el análisis de materiales Figura 2.7

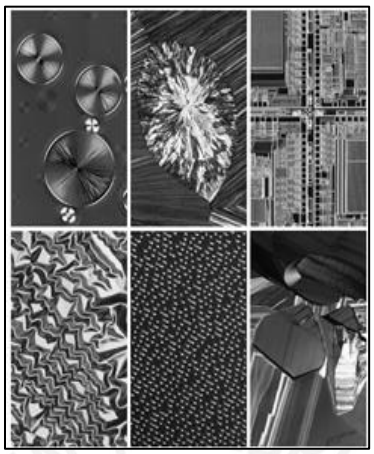


Figura 2.7 Imágenes amplificadas digitalmente para su estudio. Extraído de (Gonzalez & Woods, 2008)

- **Reconocimiento de caracteres**

  La Identificación y clasificación de las letras dentro de una determinada imagen. Un caso de uso sería la extracción de información de las personas en base a su documento de identidad Figura 2.8.

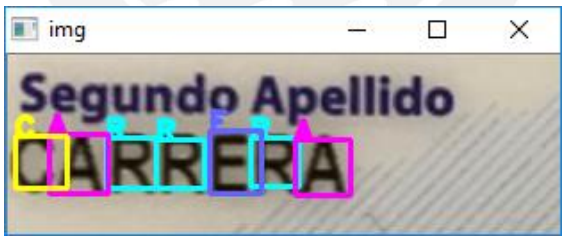


Figura 2.8 Detección y clasificación de las letras que componen el apellido de una persona. Imagen de autoría propia.

2.1.4 Conclusiones

Dado que el presente trabajo tiene como objetivos detectar y clasificar las letras en la imagen de un DNI electrónico peruano, se vio conveniente conocer primero el objeto de estudio, el DNI electrónico peruano y la información que éste contiene de una persona. Una vez conocido el DNI pasamos a estudiar el concepto básico de una imagen y cómo a través de la visión por computador los objetos son representados

por una imagen digital. Posteriormente se hace una revisión de cómo la visión por computador usa las imágenes para detectar objetos contenidos en ellas o clasificar la imagen como un todo y por último la aplicación de la visión por computador.

2.2 Estado del arte

2.2.1 Método usado en la revisión del estado del arte

En el presente capítulo se presenta la revisión sistémica del estado del arte sobre las experiencias de OCR en documentos de identidad, proponiendo un marco de referencia para investigaciones posteriores relacionados.

Las fuentes de contenido científico a usar son:

- IEEE Explore
- ACM Digital Library
- Scholar Google

Estas son fuentes de información que son reconocidas y acreditadas para la revisión y publicación de artículos científicos, lo cual le da un valor alto de confiabilidad.

2.2.1.1 Formulación de la pregunta

La pregunta de investigación formulada fue:

¿Qué técnicas de aprendizaje máquina que se han utilizado para extraer información de las personas en base a sus documentos de identidad?

La motivación de la pregunta es el identificar las técnicas de aprendizaje máquina que se han utilizado para extraer información de las personas en base a sus documentos de identidad.

Las palabras clave usadas para resolver la pregunta son: “character”, “image”, “images”, “ocr”, “recognition”, “dni”, “dnis”, “dnie”, “Identity card”, “ID card”, “classification”, “information”, “extraction”, “driving license”.

2.2.1.2 Selección de las fuentes

En la búsqueda usamos mismos criterios de búsqueda y selección para obtener investigaciones que cuenten con el mismo conocimiento especializado. Los criterios de búsqueda fueron:

Tabla 2.1 Cadenas, bloques, términos y conectores lógicos para la búsqueda.

Cadena	Bloque	Término
1	1.1	dni OR dnis OR dnie OR identity card OR id card, OR driving license
	1.2	ocr OR recognition OR classification OR “character recognition”
2	2.1	“character recognition” AND image

Los criterios de selección son:

Criterios de inclusión:

1. Se aceptan los artículos de conferencias y revistas que pertenecen a las bases de datos indexadas para el presente trabajo.
2. Se aceptan los artículos que en su contenido describan los modelos y/o técnicas de aprendizaje máquina para extraer información de documentos.
3. Se aceptan artículos con no más de 7 años de antigüedad.
4. Se aceptan artículos con modelos de clasificación de letras con una precisión mayor o igual al 70%.

Criterios de exclusión:

1. Estudios cuyos títulos y/o resúmenes no estén relacionados al tema del presente trabajo.
2. Estudios con idiomas distintos al inglés y español.
3. Artículos duplicados.
4. Artículos que traten sobre documentos personales, pero no mencionen técnicas o modelos para extraer información textual, ejemplo: firmas, rostros, huellas, solo números.

Temporalidad: se tomarán en cuenta los artículos no más de 7 años de antigüedad.

2.2.1.3 Resultados

Al ejecutar la cadena de búsqueda en las bases de datos el 13 de octubre del 2019, con filtro adicional de año de publicación, se obtuvieron los siguientes resultados:

Tabla 2.2 Resumen de resultados de la Revisión Sistémica de la Literatura

Base de datos	Cadena de búsqueda	Resultados
IEEE Xplore	(("Document Title":DNI* OR “identity card” OR “id card” OR "driving license") AND ("Document Title":ocr OR recognition OR classification OR "character recognition"))	135
	(("Document Title": "character recognition") AND ("Document Title": image))	35
ACM Digital Library	acmdlTitle:(+(DNI "identity card" "id card" "driving license") +(ocr recognition classification "character recognition"))	1
	acmdlTitle:(+"character recognition" +image)	7
Scholar Google	allintitle: ("identity card" OR "id card" OR "driving license" OR DNI OR DNIE OR DNIs) (OCR OR recognition OR classification OR "character recognition")	18
	allintitle: image "character recognition"	113

2.2.2 Investigaciones en el tema

A continuación, se presenta las investigaciones más relevantes que abordan o responden la pregunta de investigación formulada.

2.2.2.1 Fuzzy logic based vehicular plate character recognition system using image segmentation and scale-invariant feature transform

En esta investigación se propone un método de reconocimiento óptico de caracteres en placas de autos, usando una transformación de características de escala invariante con segmentación de imagen (SIFT) y lógica difusa. La segmentación de imagen separa cada carácter en una zona propia dentro de la imagen a fin de extraer las características por carácter. La transformación de características de escala invariante (SIFT) por otro lado permite la extracción de dichas características. Los algoritmos de lógica difusa toman las características y se propone a detectar los caracteres correctamente logrando una precisión de 90.75%. (Anjerome, et al., 2016)

2.2.2.2 A character recognition method in natural scene images

La presente investigación revisada presenta un método para el reconocimiento de caracteres en imágenes de escenas naturales, primero un algoritmo robusto localiza el texto en la imagen y el reconocimiento de caracteres se basa en un descriptor Histograma de gradientes orientados (HoG) y como clasificador KNN, el cual es también es comparado con otros descriptores típicamente usados para el reconocimiento de caracteres, y sus resultados muestran un buen desempeño con el método propuesto (80.4%). (González, Bergasa, Yebes, & Bronte, 2012)

2.2.2.3 Automatic OCR system for Colombian DNIs

Se presenta un sistema que permite el reconocimiento óptico de caracteres en el documento de identidad colombiano. Para ello se sigue una metodología que a través de OCR elimina el fondo holográfico y clasifica los caracteres usando Template Matching a través de la función de OpenCV: Normalized Cross-Correlation (NCC) y usando una data para test de 34 DNIs se obtuvo un desempeño del 92.6%. (Isaza, Vargas, Gaviria, & Hernández, 2012)

2.2.2.4 Character extraction in web image for text recognition

En la investigación se propone una técnica de preprocesamiento para mejorar el reconocimiento de texto en una imagen. La técnica propuesta primero suaviza la imagen para incrementar el contraste de los bordes, luego las imágenes suavizadas

son binarizadas en cada canal de color RGB. Una herramienta de OCR identifica los caracteres candidatos después de una corrección de inclinación, y se compara la precisión del modelo luego de aplicar esta técnica y otras técnicas de preprocesamiento. (Su, Lu, Phan, & Tan, 2012)

2.2.2.5 Image character recognition using deep convolutional neural network learned from different languages

En esta investigación se propone un modelo shared-hidden-layer deep convolutional neural network (SHL-CNN) para el reconocimiento de caracteres. En SHL-CNN las capas ocultas son comunes a todos los caracteres de diferentes lenguajes, realizando un proceso de extracción de características universal que enfatiza en un aprendizaje común de los rasgos existentes en los caracteres en los distintos idiomas como los trazos, mientras que la capa final de sofmax es dependiente del lenguaje y entrenado con caracteres del lenguaje indicado. La efectividad del modelo es verificada tanto en inglés como en chino para la tarea de reconocimiento de caracteres, asimismo se comparó el modelo con otro que solo había sido entrenado con caracteres de un solo lenguaje y se observó que el modelo planteado tiene un mejor rendimiento, entre 16-30%, y comparados con algunos modelos actuales un 35.7%. (J, Chen, Feng, & Xu, 2014)

2.2.2.6 An efficient character recognition method using enhanced HOG for spam image detection

Ante mensajes de SPAM con contenido de imágenes se propone debido a su atracción y al ser una imagen aumenta la dificultad para detectarlo como spam por su contenido, ante esto se propone un modelo SVM con el vector de características HIG para lograr este objetivo usando para en entrenamiento y validación las bases de datos Char74K y ICDAR2003 logrando un rendimiento de 94.2% y 86.01% respectivamente. (Naiemi, Ghods, & Khalesi, 2019)

2.2.2.7 Indonesian ID Card Recognition using Convolutional Neural Networks

Se busca extraer información del documento de identidad de Indonesia, primero se extrae el área que contiene el número del documento y este se separa dígitos, luego

se propone como modelos: SVM y CNN para inferir el valor del dígito; para detectar caracteres sobre una base de datos propia de 10000 imágenes se obtuvo una precisión de 89% y para detectar el número de documento sobre una base de datos propia de 100 documentos de identidad se obtuvo una precisión para clasificar letras de 76%. (Pratama, Satyawan, Fajar, Fikri, & Hamzah, 2018)

2.2.2.8 A System to Localize and Recognize Texts in Oriented ID Card Images

Se presenta un modelo que busca localizar y reconocer el texto en una tarjeta de identificación china, para ello primero localiza los textos con Adaboost, aplica un algoritmo para corregir la orientación del texto y por último se reconoce el texto aplicando un modelo de clasificación CNN, en este último paso se probó sobre una base de datos propia de 363210 imágenes, en tres modelos obteniendo como precisión para clasificar letras los siguientes resultados: LeNet-5 73.9%, CNN propuesto 90.4% y ResNet-56 99.7%. (Xu & Wu, 2018).

2.2.2.9 ID card identification system based on image recognition

En la investigación se busca extraer la información de la persona en base a su documento de identidad chino, para ello primero en la imagen corrige el brillo, convierte a escala de grises y luego a blanco y negro; después de ello segmenta los caracteres de cada palabra en letras y por último clasifica las letras con Template Matching para género, fecha de nacimiento, nacionalidad y número y SVM para nombre y dirección, los resultados sobre una base de datos propia de 20 documentos son 70% de precisión para clasificar letras y 90% para clasificar números. (Fang, Fu, & Xu, 2017)

2.2.2.10 Application of Convolutional Neural Network for Optical Character Recognition Designed for Kazakhstan Identity Cards

Se busca identificar las letras en el documento de identidad de Kazajistán, para ello proponen modelos CNN con diferentes estructuras para clasificar 3 grupos: caracteres de Kazajistán, caracteres de inglés, dígitos; al probar sobre una base de datos propia de 20770 imágenes de caracteres (10170 dígitos, 3456 ingles, 7144 Kazajistán) la precisión en cada grupo fue: en caracteres de Kazajistán 94.55%, en caracteres de

inglés 96.83%, en dígitos 99.18%. (Amirgaliyev, Kuatov, & Baibatyr, 2017)

2.2.2.11 FVI: An End-to-end Vietnamese Identification Card Detection and Recognition in Images

Se busca extraer las letras en el documento de identidad de Vietnam, para ello a través de un dispositivo móvil se envía la imagen del documento al servidor donde será procesarlo y devolverá los textos del documento. En el sistema se detectan las letras con un modelo basado en RetinaNet, luego se extraen las características con un modelo CNN basado en Inception-v3 y finalmente se extrae el texto con un modelo RNN; sobre una base de datos propia de 1000 imágenes, la precisión del modelo para clasificar las letras fue: número de identidad 86.7%, nombre: 85.7%, fecha de nacimiento: 79.7%, dirección: 70.24%. (Liem, et al., 2019)

2.2.2.12 Chinese Character Recognition Based on Residual Separable Convolutional Neural Network

En la investigación se busca identificar las letras en un documento de identidad de China, se utiliza un modelo propuesto Red neuronal convolucional, separable y residual (RSCNN por sus siglas en inglés), sobre una base de datos propia de 94125 imágenes (25 por cada uno de los 3765 caracteres chinos más comunes) se obtuvo una precisión para clasificar las letras de 92.73%. (Xiang, et al., 2018)

2.2.3 Productos desarrollados

En la presente sección mostraremos los productos existentes, así como su relación con el tema del trabajo y sus funcionalidades.

2.2.3.1 OmniPage de Nuance

Un excelente producto de escritorio para empresas a pesar de su precio y que es solo para Windows es capaz de trabajar con múltiples fuentes (imágenes, pdf, word) y brinda además de una muy precisa herramienta OCR, la capacidad de reconocer el formato, tablas, imágenes, márgenes y devolver no solo un texto plano sino un documento con todos los elementos presentes en la fuente. (Nuance, OmniPage Ultimate, 2019).

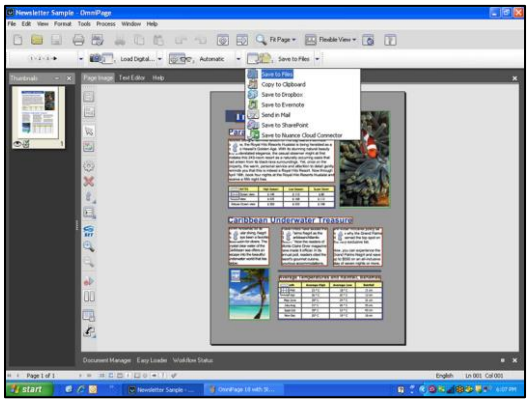


Figura 2.9 Detección de texto, tablas, imágenes y formatos. Extraído de (Nuance, OmniPage Ultimate, 2019)

2.2.3.2. Paperport de Nuance

Un buen producto de escritorio que digitaliza documentos a través su famoso algoritmo de OCR, al igual que OmniPage solo para Windows, aunque con un precio más asequible, a cambio tiene algunas limitaciones como: menor rendimiento en la detección de los caracteres, no tiene soporte para los idiomas japonés, coreano y chino, menor detección de formatos, menos rendimiento al detectar tablas, formatos, colores, variedad de tamaño de letra. (Nuance, Paperport Professional 14, 2019)

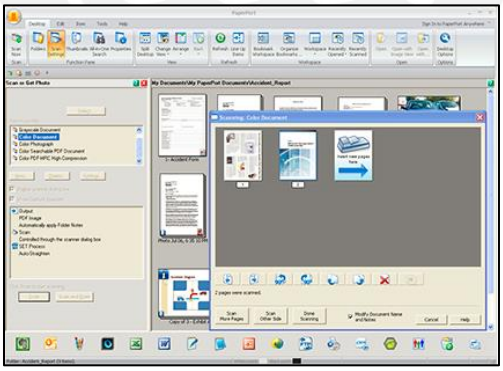


Figura 2.10 Flujo de conversión a PDF. Extraído de (Nuance, Paperport Professional 14, 2019)

2.2.3.3 Abbyy FineReader 12 Professional

Abbyy es un OCR conocido por su simplicidad y rapidez, aunque con menor precisión y funcionalidades que OmniPage, esto lo balancea con soporte para más idiomas y una interfaz más simple y amigable, reducción de ruido en las imágenes y también

cuenta con soporte para detectar tablas, imágenes y formatos. cuenta con soporte para Windows y Mac. (ABBYY, 2019)

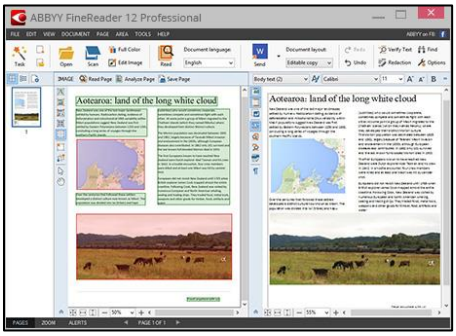


Figura 2.11 Detección de texto e imágenes. Extraído de (ABBYY, 2019)

2.2.3.4 Soda PDF Business

Soda PDF es un aplicativo para Windows, Mac, y Web, enfocado en el manejar archivos en PDF permitiendo convertir archivos word, excel, ppt, html, imagen y en sentido inverso convertir archivos en estos formatos a PDF, también hace uso de OCR para detectar caracteres en los archivos e incluso dentro de imágenes en su versión Business. (Software, 2019)

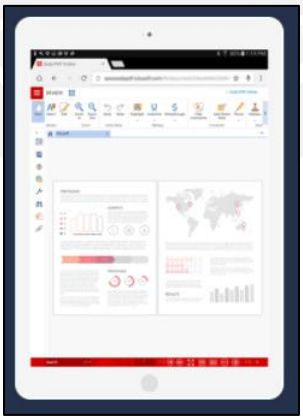


Figura 2.12 Menú principal de Soda PDF. Extraído de (Software, 2019)

2.2.3.5 ID mobile de ICAR

ID mobile es un servicio web y móvil que permite obtener los datos de un documento de identidad en 22 segundos, los datos están limitados a nombre, primer apellido, fecha de nacimiento y sexo. (ICAR, 2019)

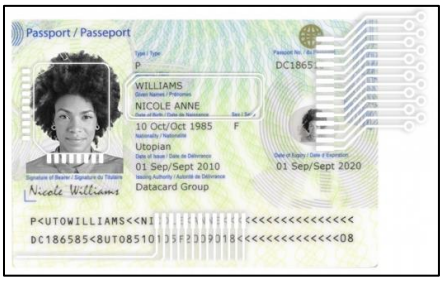


Figura 2.13 Detección de texto y foto de un documento de identidad. Extraído de (ICAR, 2019)

2.2.4 Resumen del estado del arte

A continuación, mostraremos el resumen de lo encontrado y en la siguiente sección las conclusiones

2.2.4.1 Investigaciones en el tema

Tabla 2.3 Resumen de las investigaciones seleccionadas.

Investigación	**Método de reconocimiento**	**Problema que intenta solucionar**	**Bases de datos usadas**	**Tasa de reconocimiento**
Fuzzy logic based vehicular plate character recognition system using image segmentation and scale-invariant feature transform (Anjerome, et al., 2016)	SIFT y lógica difusa	Detectar los caracteres en la placa de un vehículo.	Base de datos propio de placas de Filipinas, no se menciona cantidad de imágenes.	Precisión al reconocer caracteres: 90.75%
A character recognition method in natural scene images (González, Bergasa, Yebes,	HOG-KNN	Detectar caracteres en imágenes de calles	5482 muestras extraídas de ICDAR 2003 Robust Reading Competition	Precisión al clasificar caracteres: 80.4%

& Bronte, 2012)				
Automatic OCR system for colombian DNIs (Isaza, Vargas, Gaviria, & Hernández, 2012)	Template Matching	Detectar los caracteres presentes en el DNI colombiano	34 muestras propias	Precisión al clasificar caracteres: 92.6%
Character extraction in web image for text recognition (Su, Lu, Phan, & Tan, 2012)	ABBYY OCR	Detectar caracteres en imágenes de páginas web, se enfoca en mejorar el preprocesamiento	918 imágenes tomadas de (ICDAR) 2011, Robust Reading Competition in Born-Digital Images	Precisión al clasificar caracteres: 72.33%
Image character recognition using deep convolutional neural network learned from different languages (J, Chen, Feng, & Xu, 2014)	SHL-CNN	Detectar caracteres de diferentes idiomas (inglés-chino).	ICDAR 2003 competition (train: 6185 imágenes, test: 5430 imágenes y muestra: 854 imágenes) para inglés. 50 horas de vídeo de noticieros de TV china (5036 líneas de texto) para chino	-
An efficient character recognition method using enhanced HOG for spam image	SVM-HOG	Detectar caracteres en imágenes de SPAM.	Char74K (75776 caracteres entre a-z, A-Z y 0-9) y ICDAR2003 (train: 6185 imágenes, test:	Precisión al clasificar caracteres: En Char74K 94.2% y En ICDAR2003

detection (Naiemi, Ghods, & Khalesi, 2019)			5430 imágenes y muestra: 854 imágenes)	86.01%
Indonesian ID Card Recognition using Convolutional Neural Networks (Pratama, Satyawan, Fajar, Fikri, & Hamzah, 2018)	CNN y SVM	Detectar letras en el documento identidad de Indonesia.	Base de datos propio de 10000 caracteres y 100 documentos (para validar la clasificación del número).	Precisión al clasificar caracteres: CNN: 89% SVM: 71% Precisión al identificar el número: CNN: 76%
A System to Localize and Recognize Texts in Oriented ID Card Images. (Xu & Wu, 2018)	Para detección de caracteres: Adaboost Para clasificación de caracteres: LeNet-5, CNN propuesto y ResNet-56.	Detectar caracteres en una tarjeta de identidad de China.	Base de datos de 363210 imágenes sintéticas propias (3562 clases, 3500 caracteres chinos, 52 caracteres en mayúsculas y minúsculas y 10 dígitos).	Precisión al clasificar caracteres: LeNet: 73% CNN Propuesto: 90.4% ResNet-56: 99.7%
ID card identification system based on image recognition. (Fang, Fu, & Xu, 2017)	Template matching para caracteres en: genero, fecha de nacimiento, nacionalidad y número SVM para nombre y dirección	Extraer información de una persona a partir de su tarjeta de identidad china.	Base de datos propia de 20 tarjetas de identidad china.	Precisión al clasificar caracteres: 70% Precisión al clasificar dígitos: 90%

Application of Convolutional Neural Network for Optical Character Recognition Designed for Kazakhstan Identity Cards. (Amirgaliyev, Kuatov, & Baibatyr, 2017)	CNN	Se busca identificar las letras en el documento de identidad de Kazajistán.	Base de datos propia de 20770 imágenes de caracteres (10170 dígitos, 3456 ingles, 7144 Kazajistán)	Precisión en cada grupo fue: Caracteres de Kazajistán 94.55% Caracteres de inglés 96.83% Dígitos 99.18%
FVI: An End-to-end Vietnamese Identification Card Detection and Recognition in Images. (Liem, et al., 2019)	Detección de texto: RetinaNet Extracción de características: CNN Inception-v3 Reconocimiento de texto: RNN	Se busca identificar las letras en el documento de identidad de Vietnam.	Base de datos propia de 1000 imágenes de documentos de identidad de Vietnam.	Precisión al detectar texto: Número de identidad: 86.7% Nombre: 85.7% Fecha de nacimiento: 79.7% Dirección: 70.24%
Chinese Character Recognition Based on Residual Separable Convolutional Neural Network. (Xiang, et al., 2018)	Red neuronal convolucional, separable y residual RSCNN	Se busca identificar las letras en el documento de identidad de China.	Base de datos propia de 94125 imágenes, 25 por cada uno de los 3765 caracteres chinos más comunes.	Precisión para detectar caracteres: 92.73%

2.2.4.2 Productos desarrollados

Tabla 2.4 Resumen de los productos analizados.

Producto	**Plataforma**	**Características**
OmniPage de Nuance (Nuance, OmniPage Ultimate, 2019)	Windows	OCR en documentos, con formato y múltiples lenguajes
Paperport de Nuance (Nuance, Paperport Professional 14, 2019)	Windows	OCR en documentos, con formato
Abbyy FineReader 12 Professional (ABBYY, 2019)	Windows,Mac	OCR en documentos, con formato y múltiples lenguajes
Soda PDF Business (Software, 2019)	Windows, Mac, Web	OCR en documentos, con versión web
ID mobile de iCar (ICAR, 2019)	Android, iOS, Servicios Web-SOAP	OCR en documentos de identidad
ABBYY Business Card Reader (ABBYY, 2019)	Android, iOS	OCR en documentos de identidad

Al evaluar el aplicativo ABBYY BCR en un DNI electrónico peruano estos fueron los resultados:

Tabla 2.5 Resultado de usar ABBYY BCR en un DNI electrónico peruano

Campo	Valor real	Reconocido	Acierto
Nombre	Tomas Ivan	Tomas	50%
Apellidos	Carrillo Fuertes	Ivan	0%

2.2.4.3 Servicios en la nube desarrollados

A continuación, presentamos los servicios de visión desarrollados en la nube:

Tabla 2.6 Resumen de los servicios en la nube desarrollados

Servicio	**Plataforma**	**Características**
API Vision Detección de texto en documentos (Google, 2019)	Google Cloud Platform https://cloud.google.com	OCR en documentos, datos anónimos, no detecta números. USD $1.5 por mil consultas. Demora aproximadamente 30 s.
Amazon Rekognition Text in Image (Amazon, 2019)	Amazon Web Services https://aws.amazon.com	OCR en documentos, datos anónimos, sí detecta números, USD $1.0 por mil consultas. Demora aproximadamente 20 s.
Computer Vision API v2.0 OCR (Microsoft, 2019)	Azure https://azure.microsoft.com	OCR en documentos, datos anónimos, sí detecta números. USD $1.5 por mil consultas. Demora aproximadamente 2 s. leyendo imagen desde una URL.
Visual recognition Text Model (IBM, 2019)	Watson https://www.ibm.com/watson/services/visual-recognition	OCR en documentos, Datos anónimos, sí detecta números. USD $2.0 por mil consultas. Demora aproximadamente 25 s.

Los servicios en la nube desarrollados tienen las siguientes características:

- Detectan todas las letras de la palabra.
- La clasificación de las letras es relativamente rápida, de 2 a 5 segundos todo el documento.
- Bajo costo aproximadamente USD $1.5 por mil imágenes.
- Procesan 1 imagen a la vez.
- Tiempo de carga de una imagen (2.87 MB) para ser analizada de 10 a 20 s.
- Para procesar 30 imágenes capturadas en 1 segundo tomaría aproximadamente de 20 a 30 s. usando llamadas en paralelo.
- Depende de la existencia de una conexión a internet.
- Modelos de OCR propios del servicio que no se pueden modificar.

2.2.5 Conclusiones sobre el estado del arte

A continuación, presentamos las conclusiones derivadas de revisar las investigaciones que buscan responder nuestra pregunta de investigación:

- La revisión sistemática nos ayuda a explorar los trabajos que han abordado el tema de interés expresado en la pregunta de investigación, obtener evidencias sólidas y variadas en cada fuente, a pesar de haber usado los mismos términos de búsqueda.
- Las investigaciones han abordado el reconocimiento de caracteres en diferentes idiomas y en diferentes contextos, por ejemplo, en: documentos de identidad, placas de auto, imágenes web e imágenes naturales.
- Se observa una mayor precisión en los modelos cuya fuente son imágenes con menor variación de color, fuente, tamaño (placas y documentos de identidad).
- Para la clasificación de caracteres en documentos de identidad se usaron los modelos: ResNet-56, CNN, Template Matching, RNN, LeNet-56 y SVM.
- En cada trabajo se evidencia la importancia del preprocesamiento de las imágenes para optimizar el entrenamiento del modelo.

Conclusiones de la revisión de los productos que brindan directa o indirectamente soluciones parciales al problema de reconocimiento de caracteres:

- Los productos de Nuance (OmniPage, Paperport), presentan soluciones muy completas para la digitalización y reconocimiento de caracteres incluyendo sus fuentes, tamaño y color, pero presentan su limitación de plataforma, solo Windows.
- Otros productos han surgido en el mercado como Abbyy y Soda PDF que también presentan soluciones OCR, que además reconocen fuentes, tamaño y color; incluyen las plataformas: Mac y Web; pero con menor precisión que los productos de Nuance.
- El producto ID Mobile de ICAR presentan una solución OCR en línea que permite reconocer información muy básica del DNI, solo el nombre, primer apellido, sexo, fecha de nacimiento, y tiene un tiempo de procesamiento de aproximadamente 22 segundos.

Conclusiones de la revisión de los servicios en la nube que brindan directamente soluciones parciales al problema de reconocimiento de caracteres:

- Todos los servicios analizados detectan los caracteres de cada palabra, pero de forma anónima, es decir no reconoce a que dato de la persona corresponde esta palabra. Para un aplicativo que debe presentar los datos de las personas no aplicaría.
- El tiempo de carga para procesar 1 imagen es muy alto de 10 a 20 s.
- Para un aplicativo móvil que debe trabajar sin conexión a internet (offline) no aplicarían pues estos servicios requieren internet.
- Al momento de la presente revisión bibliográfica, el servicio de Google Cloud Platform (GCP) no detecta números

3 Modelo para la segmentación de palabras y letras

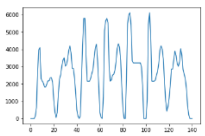
3.1 Introducción

En el presente trabajo empezaremos con la detección de las palabras y letras de un dato personal del DNI electrónico, con ello reduciremos la dimensión de características de la imagen a ser procesada, al tener menos características reducimos el número de cálculos, por lo que tendremos un modelo clasificador menos complejo y más rápido para clasificar letras. Para alcanzar este objetivo se ha experimentado con diferentes técnicas, librerías y algoritmos. El presente capítulo presentará las técnicas empleadas para segmentar las palabras y letras de cada dato personal en el DNI electrónico.

3.2 Descripción

La segmentación de palabras y letras de un dato personal es un primer paso para la reducción de características, a su vez permite agrupar aquellas letras que conforman un dato de la persona (ejemplo: primer apellido) y luego cada letra sea clasificada. En la Tabla 3.1 se muestran las etapas.

Tabla 3.1 Etapas para la segmentación de palabras y letras.

Etapa	**Descripción**	**Ejemplo**
Adquisición	Lectura de la imagen del DNI	
Localización	Localizar solo la región del dato	
Pre-procesado	Filtro de intensidad de color	
	Eliminar ruido	
Detección	Detección de espacios entre las letras y palabras del dato.	

3.2.1 Adquisición

La etapa de adquisición, es el primer paso de nuestro modelo, donde la imagen a ser evaluada es leída y convertida en un formato digital de pixeles con tres canales de color (BGR) y una intensidad variante entre 0 y 255, en OpenCV las imágenes son tratados como matrices.

La forma de implementar es con la librería openCV y utilizamos la función “imread” descrita en el Código 3.1; se debe tomar en cuenta que openCV maneja los canales de colores BGR, el cual es diferente al estándar RGB (de los colores en inglés: Red, Green, Blue):

opencv2.imread(string src, int flag)
donde:
src es la dirección relativa o absoluta de la imagen
flag es como la imagen debería ser leída:

- cv2.IMREAD_COLOR Lee la imagen a color sin tomar en cuenta la transparencia
- cv2.IMREAD_GRAYSCALE Lee la imagen en escala de grises
- cv2.IMREAD_UNCHANGED Lee la imagen como e, incluyendo canal alpha.

Código 3.1 Función para leer imagenes

El resultado de esta etapa es la imagen del DNIe digitalizada que servirá de insumo para las siguientes etapas. Un ejemplo se puede ver en la Figura 3.1.

Figura 3.1 Imagen del DNIe digitalizado, resultado de la etapa adquisición.

3.2.2 Localización

En la etapa de localización ubicamos las regiones donde se encuentras los datos personales, que están prefijados en el DNI electrónico. Utilizamos solo las regiones que contienen los datos personales para reducir los costos computacionales de procesar la imagen completa del DNI.

Para ello utilizamos la función submat descrita en Código 3.2, submat devuelve una región de una imagen.

```
Mat.submat(int rowStart, int rowEnd, int colStart, int colEnd)
donde:
Mat es la matriz que representa la imagen
rowStart es la fila donde inicia la región
rowEnd es la fila donde termina la región
colStart es la columna donde inicia la región
colEnd es la columna donde termina la región
```

Código 3.2 Función submat, devuelve una región de una imagen.

El resultado esperado de esta etapa es una región de la imagen que contiene un dato de la persona. Un ejemplo del resultado se puede apreciar en la Figura 3.2

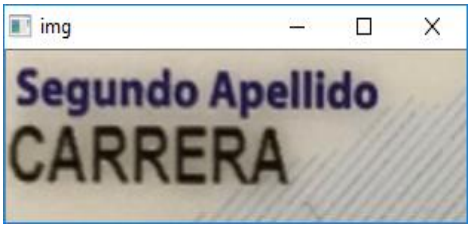


Figura 3.2 Imagen de un dato de la persona, resultado de la etapa de localización.

3.2.3 Pre-procesado

En la etapa de preprocesamiento tratamos la imagen para separar los caracteres del fondo y el ruido. Aplicando un filtro por color obtenemos una máscara que representa el dato en blanco y negro, aplicamos la técnica de “opening”, opening consiste en aplicar erosión seguido de dilatación sobre una imagen para eliminar el ruido.

El primer paso de esta etapa es importante pues permite la reducción de la dimensionalidad de 3 a 1 y separa las letras del fondo. Para esto usaremos la librería opencv con la función “cv2.inRange” descrita en Código 3.3.

cv2.inRange(imagen, lower_range, upper_range) donde:
imagen es el objeto imagen a ser filtrada
lower_range es el mínimo valor de los pixeles a ser filtrados, ejemplo: (0,0,0)
upper_range es el máximo valor de los pixeles a ser filtrados, ejemplo: (50,50,50)

Código 3.3 Descripción de la función inRange.

El segundo paso de esta etapa es igual de importante pues permite darle tolerancia al modelo frente a imágenes con ruido, pues elimina el ruido. Para ello utilizaremos el método “opening” a través de la función “cv2.morphologyEx” descrita en Código 3.4.

cv2.morphologyEx(imagen, mode, kernel)

donde:

imagen es el objeto imagen a procesar

mode es el método a emplear, ejemplo: cv2.MORPH_OPEN (opening)

kernel es el kernel a usar en el método, ejemplo ([[0,1,0],[1,0,1],[0,1,0]])

Código 3.4 Descripción de la función morphologyEx

El resultado de esta etapa es una imagen del dato con una clara diferenciación entre el fondo y las letras, así como menos ruido por utilizar el método de “opening”. Un ejemplo del resultado se puede apreciar en la Figura 3.3.

Figura 3.3 Imagen de un dato de la persona con contraste entre letras y fondo, resultado de la etapa de preprocesamiento.

3.2.4 Detección

En esta etapa procedemos con la detección de las palabras y letras del dato personal. Cada letra detectada será posteriormente clasificada. Para ello calcularemos el promedio de intensidad de los pixeles de cada columna de la imagen y luego usando un algoritmo detectaremos según la intensidad el inicio-fin de cada letra y palabra. Adicionalmente calcularemos el promedio de intensidades de cada fila para calcular los límites verticales del texto en el dato.

El resultado de esta etapa visualmente es un histograma con el promedio de intensidades de cada columna de la imagen (Figura 3.4) y un arreglo que contiene cada palabra, cada palabra es un arreglo que contiene letras y cada letras contiene sus coordenadas de inicio y fin (Figura 3.5).

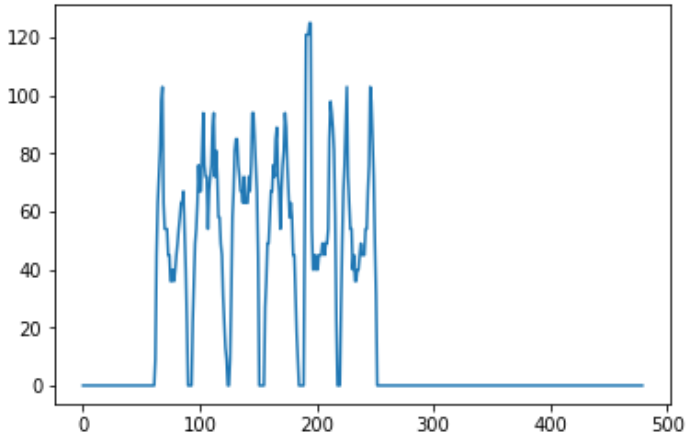


Figura 3.4 Histograma con el promedio de intensidades de cada columna, resultado de la etapa de detección.

```
[[[62, 90], [94, 124], [126, 151], [156, 185], [190, 218], [221, 252]]]
nro palabras:1
nro letras:6
```

Figura 3.5 Arreglo de palabras y letras, resultado de la etapa detección

4 Modelo para la clasificación de letras

4.1 Introducción

En el presente capítulo se describe el proceso seguido para la generación, entrenamiento, refinamiento y validación de los modelos de clasificación.

Para realizar la clasificación en un entorno móvil con pocos recursos se optó por dos modelos: Adaboost y YOLO v3 Tiny. Según se observa en una de las conclusiones de la revisión del estado del arte los mejores resultados se obtuvieron con modelos CNN, por ello se propone el modelo YOLO v3 tiny que es un modelo CNN con muy buena precisión y de bajo costo computacional (Redmon & Farhadi, YOLO: Real-Time Object Detection, 2019), también se propone Adaboost que es un modelo binario basado en arboles de decisiones y es un modelo muy ligero, Para el presente trabajo se observó que el modelo de Adaboost pesa en promedio 1.12 KB estando compuesto por 2 árboles, y es super rápido al realizar la clasificación, en un video de 30 FPS (del inglés “frame per second”) logra realizar las 30 clasificaciones dentro del segundo. Por otro lado, YOLO v3 tiny es un modelo de red neuronal profunda ligera que logra obtener mejores que resultados que Adaboost, pero es más pesado, para el presente trabajo se observó que pesa 34118 KB, y es más lento, en un video de 30FPS logra clasificar solo 2 imágenes de las 30 en un segundo.

4.2 Adaboost

Para el entrenamiento del modelo Adaboost necesitaremos imágenes negativas (no contienen el objeto a detectar) y un vector que contendrá las imágenes positivas (sí contienen el objeto a detectar). Por ello como primer paso creamos el vector de imágenes positivas y luego entrenamos el modelo.

4.2.1 Pasos para generar vector de muestras positivas

Para crear el vector de imágenes, primero crearemos imágenes positivas sintéticas del objeto. Dada la naturaleza de los objetos a detectar, caracteres que utilizan una misma fuente y tamaño de letra, color, tipo de línea, y están en mayúsculas, y que se cuenta con pocas muestras positivas, se ha optado por utilizar la herramienta de OpenCV llamada “opencv_createsamples” que en base a una imagen del objeto crea

nuevas imágenes positivas (imágenes que contienen el objeto a detectar). Para lograr esto la herramienta usa imágenes negativas (imágenes que no contienen el objeto a detectar) como fondo y le inserta la imagen del objeto ya distorsionado, rotado y con ruido, la imagen resultante es una nueva imagen positiva. Un ejemplo se ve en la Figura 4.1.

Figura 4.1 A la izquierda imagen real del objeto, al centro imagen negativa que no contiene el objeto y a la derecha una imagen positiva sintética.

Primero usaremos la función opencv_createsamples como se muestra en el Código 4.1 para crear las nuevas imágenes positivas y las listaremos en el archivo "info.lst".

```
opencv_createsamples -img img/a1.jpg -bg bg.txt -info info/info.lst -pngoutput info -maxxangle 0.1 -maxyangle 0.1 -maxzangle 0.1 -num 1848 -bgcolor 0 -bgthresh 0
```

Código 4.1 Sentencia para crear 1848 imágenes positivas a partir de la imagen a1.jpg

Los parámetros de la función son:

- **img** como imagen positiva original
- **bg** lista de imágenes de fondo
- **info** información de las imágenes positivas creadas
- **pngoutput** generación de las imágenes positivas
- **max(x/y/z)angle** ángulos de distorsión para las imágenes sintéticas a crear
- **num** numero de imágenes sintéticas a crear
- **bgcolor** +- **bgthresh** rango de colores a ser considerados transparentes

En el Código 4.2 creamos el vector "positives.vec" usando 1848 imágenes al azar de "info.lst" y las guardamos con una dimensión de 12 x 12 pixeles.

```
opencv_createsamples -info info/info.lst -num 1848 -w 12 -h 12 -vec positives.vec
```

Código 4.2 Sentencia para crear el vector que contendrá las imágenes del objeto

4.2.2 Pasos para entrenar el modelo y detectar

Luego de generar el vector con las imágenes positivas se procede al entrenamiento del modelo Adaboost a través de la función “opencv_traincascade” de openCV. Para el refinamiento del modelo se adicionaron nuevas muestras positivas con los falsos negativos, junto con la modificación de los hiperparámetros para modificar la forma como se entrena el modelo. Para la predicción se usará la función “detectMultiScale” del modelo entrenado para clasificar una imagen como una letra, esta función calculará las características de Haar y clasificará la imagen como una letra.

Como se muestra en el Código 4.3 entrenamos el modelo con descriptor HAAR usando 1400 positivos de vec y 700 negativos de bg.txt, marco de entrenamiento 12 x 12 y stages 10.

```
opencv_traincascade -data data -vec positives.vec -bg bg.txt -numPos 1400 -
numNeg 700 -numStages 10 -w 12 -h 12
```

Código 4.3 Entrenamiento del modelo usando 1400 imágenes positivas y 700 imágenes negativas.

Para la detección como se muestra en el Código 4.4 usamos la función detectMultiScale del modelo ya entrenado y le pasamos la imagen.

```
model.detectMultiScale(obj imagen)
Donde:
imagen: es la imagen donde el modelo “model” detectará un objeto determinado
```

Código 4.4 Detección de una letra en una imagen

Resultado:

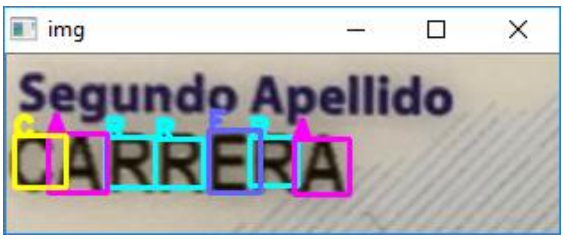


Figura 4.2 Detección de letras A, C, E y R en una zona de información del apellido materno.

4.3 YOLO v3 tiny

Dada la naturaleza de YOLO al ser una red neuronal profunda, al entrenar con imágenes sintéticas se obtiene un pobre rendimiento, por lo que se optó por usar 78000 imágenes obtenidas a partir de 22 videos propios de DNI electrónicos, para el entrenamiento se usarán 52000 imágenes, 2000 imágenes de cada letra y para la validación 26000 imágenes, 1000 imágenes por letra.

Para realizar la clasificación “YOLO v3” requiere de 4GB GPU-RAM al ser una capacidad superior a la de un smartphone promedio, se optó por “YOLO v3 Tiny” que requiere solo 1 GB GPU-RAM. Para obtener las imágenes de entrenamiento se grabaron los DNI de prueba dejando mostrar en la pantalla los resultados del modelo de detección de letras y palabras, como se muestra en la Figura 4.3.

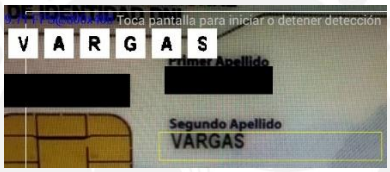


Figura 4.3 Obtención de imagenes de entrenamiento y validación.

Luego cada video fue separado en imágenes, se eliminaron las imágenes que no eran válidas para el entrenamiento/validación tomando como criterio imágenes muy borrosas, con poca iluminación o demasiado inclinadas; y por último de cada imagen se extrajo las imágenes de letras y estas se etiquetaron para ser usadas en YOLO.

Para el entrenamiento se usó el Código 4.5 donde se usó el archivo de configuración “yolov3-tiny-obj.cfg” el cual se puede ver completo en el anexo y que indica principalmente 26 clases (una para cada letra), la dimensión de la imagen de entrada en 96x96 al ser el mínimo común múltiplo de: 32 por requerimiento de YOLO y 48 que es la dimensión que devuelve el modelo de detección de palabras y letras del capítulo 3; se usaron los pesos iniciales: yolov3-tiny.conv.15 disponibles en la página web de YOLO.

```
darknet.exe detector train data/obj.data yolov3-tiny-obj.cfg yolov3-tiny.conv.15
```

Código 4.5 Sentencia para entrenar YOLO

5 Aplicativo móvil para extraer datos personales

5.1 Introducción

En este capítulo se tratará las etapas empleadas para la extracción de las letras de un dato personal, su clasificación y presentación.

5.2 Descripción

Para llevar a cabo la extracción de datos personales del DNI electrónico se realizarán tres etapas, la primera es la segmentación del dato personal en imágenes de letras, la segunda etapa es clasificar la imagen en una letra y la última es presentar el valor inferido del dato personal.

Tabla 5.1 Etapas para obtener las características de Haar de los caracteres pertenecientes a una zona de información.

Etapa	**Descripción**	**Ejemplo**
Extracción	Extraer la letra del dato	
Clasificación	Clasificar imagen en letra	Letra “O”
Presentación	Presentar los datos	Segundo Apellido: Carrera

5.2.1 Extracción

Para la extracción se aplicará primero el modelo de detección de palabras y letras sobre la imagen del dato personal, una vez identificados la ubicación de sus letras se proceden a extraer e insertar en un fondo blanco de 48 x 48.

5.2.2 Clasificación

Para la clasificación de una letra se usarán 100 capturas consecutivas de la imagen de la letra, cada una se clasificará usando los modelos de Adaboost. Al finalizar las 100 clasificaciones la clasificación final será la letra que más apareció en las 100 clasificaciones.

5.2.3 Presentación

Para la presentación de los resultados se usará el resultado de la clasificación y la detección de palabras y letras, para ordenar las letras del dato.

Resultado

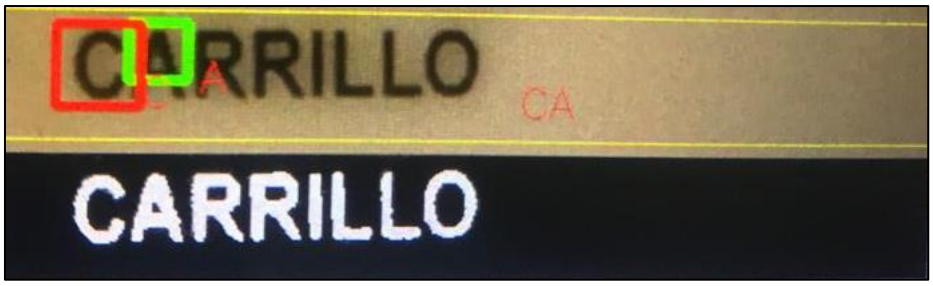


Figura 5.1 Resultado de utilizar el filtro de intensidad.

Figura 5.2 Resultado de aplicar el modelo detección y clasificación.

6 Experimentación y Resultados

6.1 Introducción

En el presente capítulo se describe el protocolo de experimentación para el modelo de segmentación de las palabras y letras, modelo de clasificación de letras y el aplicativo móvil para extraer información del DNI.

6.2 Experimentación y resultados del modelo de segmentación de palabras y letras

6.2.1 Observaciones

Para la segmentación de palabras y letras se observó que:

- Para tener letras claras cada dato personal del DNI debería tener una altura mínima de 45 pixeles.
- Para ubicar los datos personales se optó por colocar en la cámara marcos que representen cada dato personal y así el usuario se guie para ubicar el DNI en la cámara.
- Al utilizar la cámara para obtener la imagen del DNI usualmente los primeros 2 segundos esta está desenfocada y es borrosa, ante ello podríamos obviar estos segundos de la captura de la imagen o indicar cuando iniciar el proceso.
- Para no reducir el rendimiento el entorno de captura de la imagen debe tener buena iluminación, el documento debe estar sin mica o protector y el documento debe estar alineado con los marcos de los datos personales.
- Para separar las letras individualmente se utilizan histogramas que medirán la intensidad de los pixeles proyectados sobre el eje horizontal y así identificar los espacios vacíos entre letras y entre palabras.
- La cámara puede ajustar de forma automática el foco de la cámara, lo cual hace que las letras del documento no siempre sean claras.

6.2.2 Experimentación en el entrenamiento

Dado el comportamiento de las cámaras de celular, de autoenfoque, se optó por no hacer la segmentación de palabras y letras con una sola imagen sino en función de varias imágenes, siguiendo los siguientes pasos:

1. Recolectar imagen
2. Extraer zona de interés de la imagen
3. Realizar el preprocesamiento en la zona de interés
4. Aplicar el modelo de segmentación sobre la imagen preprocesada
5. Guardar el resultado del modelo de segmentación: cantidad de palabras y letras
6. Repetir desde el paso 1 al paso 5, 100 veces
7. Calcular entre los 100 resultados: cantidad de palabras con mayor frecuencia y cantidad de letras con mayor frecuencia.
8. Presentar la cantidad de palabras y cantidad de letras por palabra

6.2.2 Resultados

Se presentan los resultados al aplicar el modelo de segmentación de palabras sobre 8500 imágenes de 85 datos del documento (100 imágenes por dato). En ambos indicadores se considera una detección correcta si de una muestra de 100 imágenes la cantidad inferida con mayor frecuencia es igual al valor real.

Indicador 1 número de palabras en cada dato detectados correctamente

Indicador 1 = P / TP

P: detecciones correctas de la cantidad de palabras
TP: detecciones totales de la cantidad de palabras

Cantidad de datos 85, cantidad de imágenes por dato: 100

Indicador 1 = 85/85 = 100%

Indicador 2 número de letras en cada palabra detectados correctamente

Indicador 2 = L / TL

L: detecciones correctas de la cantidad de letras
TP: detecciones totales de la cantidad de letras

Muestra de palabras: 101, cantidad de imágenes por palabra: 100

Indicador 2 = 100/101= 99.01%

6.3 Experimentación y resultados del modelo de clasificación Adaboost

6.3.1 Observaciones

En la clasificación de letras con Adaboost se observó que:

- Para obtener una buena clasificación de las letras, es mejor aplicar el modelo de clasificación sobre cada letra por separado.
- Para una buena clasificación se usó ventanas de detección de desde 20x20 hasta 40x40 con una ratio de escalamiento de 1.2 y al menos 2 de vecinos.
- El rendimiento de un modelo de clasificación al evaluar una imagen estática es diferente al evaluar imágenes capturadas de una cámara de celular.

6.3.2 Experimentación en el entrenamiento

Dada la naturaleza de los objetos a detectar, caracteres que utilizan una misma fuente, tamaño de letra y color se siguieron los siguientes pasos para crear los modelos Adaboost que detectará cada letra:

1. Por cada letra se creó 1 imagen positiva.
2. Se extrajo las imágenes de letras de 17 DNI electrónicos reales para ser usados en la etapa de validación de los modelos.
3. Por cada letra se generó 3150 nuevas imágenes positivas distorsionando las del paso 1.
4. Se agruparon las imágenes positivas de 3 letras.
5. Por cada letra en el grupo, se entrenaron modelos Adaboost con diferentes configuraciones. El modelo tomó sus imágenes como positivas y las imágenes de las demás letras en el grupo como negativas.
6. Si un modelo tiene un buen nivel de detección (mayor a 80%), entonces se continuó con otra letra, caso contrario se repetía desde el paso 3, añadiendo una letra más al grupo.
7. Se hizo una validación por cada letra en el aplicativo móvil para validar su ratio de detección en videos, para ello se consideró las 10 primeras detecciones y sí detectaba en más de 5 se consideraba un modelo terminado y se proseguía con otro modelo.

Durante la experimentación se observó que para entrenar un modelo que detecte una sola letra se requería en promedio 5 letras en un grupo. En detalle de cada grupo de entrenamiento se presenta en la siguiente tabla 6.1.

Tabla 6.1Grupos de letras para entrenamiento

Letra a detectar	Grupo
A	ACMEID
B	ABRO
C	ACMEI
D	ACMEID
E	ACMEI
F	FG
G	FG
H	JVDOH
I	ACMEI
J	JVDOH
K	KNSLTI
L	KNSLTI
M	ACMEI
N	KNSLTI
O	CDOQ
P	NPUFG
Q	CDOQ
R	ABRO
S	KNSLTI
T	KNSLTI
U	NPUFG
V	JVDOH
W	JVDOHW
X	JVDOHX
Y	JVDOHXY
Z	JVDOHYZ

6.3.3 Resultados

Los resultados al aplicar la clasificación de letras en 67100 imágenes de 671 letras (100 imágenes por letra) de prueba se obtuvieron los siguientes resultados:

Indicador 3 letras clasificadas correctamente

Indicador 3 = C / T

C: clasificaciones correctas (Se considera correcta si de una muestra de 100 imágenes la letra inferida con mayor frecuencia es igual al valor real)
T: clasificaciones totales

Indicador 3 = 594/671 = 88.52%

6.4 Experimentación y resultados del modelo de clasificación YOLO v3 tiny

6.4.1 Observaciones

En la clasificación de letras con YOLO se observó que:

- A diferencia de Adaboost al entrenar con imágenes sintéticas se obtenía un pobre rendimiento comparado si se entrenaba con imágenes reales.
- Al entrenar con una dimensión grande (416x416) demora más al clasificar que con una menor dimensión (96x96), 2 segundos en el primer caso y 0.5 en el segundo.
- Cuando la imagen usada para el entrenamiento/validación tiene un ancho menor a 14, tiende a bajar la precisión de detección para dicha letra, por ello se decidió que el ancho mínimo sería 14.

6.4.2 Experimentación en el entrenamiento

Dado que YOLO v3 requiere aproximadamente 4 GB para poder ser ejecutado se optó por YOLO v3 tiny que requiere solo 1GB de memoria. Se siguieron los siguientes pasos para crear el modelo de YOLO tiny v3 que detectará cada letra:

1. Usando las imágenes de DNI se generaron videos de los DNI.
2. De cada video se fragmento en imágenes y de cada imagen se extrajo las imágenes de letras para el entrenamiento y validación.
3. Por cada letra se generó 3000 imágenes reales, 2000 fueron usadas para entrenamiento y 1000 para validación.
4. Se entreno el modelo usando la arquitectura de YOLO v3 tiny y sus pesos pre entrenados para clasificar 26 letras usando las imágenes del paso 2. Cada 1000 iteraciones se grabó los pesos.
5. Se hizo una validación usando las imágenes del paso 2, por peso obtenido del paso 3 para seleccionar el mejor con mayor mAP (mean average precision media de precisión promedio por sus siglas en inglés).

Al realizar el entrenamiento se usaron 52000 iteraciones, en la Figura 6.1 se aprecia en azul la función de perdida y en rojo la precisión del modelo durante el entrenamiento.

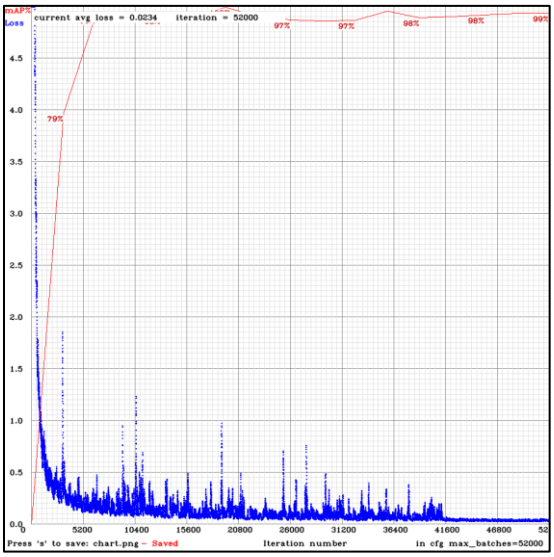


Figura 6.1 Función de Perdida y precisión durante el entrenamiento. Imagen de autoría propia.

Al validar el modelo con los pesos de cada 1000 iteraciones, se calculó la precisión, mAP así como otras métricas que se muestran en la Tabla 6.2.

Las métricas medidas en la validación son:

- **IoC:** (Intersect over union, intersección sobre unión por sus siglas en inglés)**.**
- **Precisión:** la precisión del modelo.
- **Recall:** exhaustividad del modelo.
- **F1:** f1 score del modelo
- **TP:** Verdadero Positivo.
- **FP:** Falso Positivo.
- **FN:** False Negativo

Tabla 6.2 Resultados de entrenar el modelo YOLO v3 tiny

Epoca (miles)	mAP	IoU	Precisión	Recall	F1	TP	FP	FN
1	0.558	41.13	0.59	0.43	0.50	11239	7797	1476
2	0.876	55.53	0.76	0.84	0.8	21923	6891	4077
3	0.286	20.36	0.35	0.28	0.31	7281	1346	1871
4	0.970	70.85	0.95	0.95	0.95	24797	1385	1203
5	0.990	74.24	0.92	0.99	0.96	25751	2103	249
6	0.973	77.04	0.99	0.83	0.9	21537	181	4463
7	0.991	79.34	0.98	0.99	0.98	25695	636	305
8	0.964	79.62	0.99	0.9	0.95	23522	237	2478
9	0.869	59.37	0.91	0.92	0.92	23949	2306	2051
10	0.711	48.66	0.78	0.79	0.78	20465	5709	5535
11	0.983	73.62	0.97	0.98	0.97	25392	855	608
12	0.977	78.14	0.96	0.98	0.97	25402	1049	598
13	0.987	71.5	0.96	0.97	0.97	25250	1053	750
14	0.978	66.81	0.91	0.98	0.95	25467	2388	533
15	0.984	70.63	0.98	0.99	0.98	25642	476	358
16	0.994	81.38	0.99	0.98	0.99	25599	165	401

17	0.985	75.42	0.97	0.98	0.98	25569	864	431
18	0.969	78.58	0.98	0.96	0.97	24909	568	1091
19	0.984	79.21	0.99	0.97	0.98	25126	286	874
20	0.969	80.88	1.00	0.98	0.99	2541	81	584
21	0.974	65.99	0.94	0.94	0.94	24394	1534	1606
22	0.987	79.43	0.97	0.99	0.98	25798	715	202
23	0.978	77.57	0.96	0.98	0.97	25492	927	508
24	0.984	77.29	0.98	0.98	0.98	25464	399	536
25	0.991	81.51	0.96	1.00	0.98	25873	1041	127
26	0.993	78.19	0.97	1.00	0.99	25922	692	78
27	0.984	69.54	0.99	0.98	0.98	25421	380	579
28	0.974	74.68	0.98	0.98	0.98	25401	449	599
29	0.999	76.77	0.99	1.00	1.00	25920	170	80
30	0.969	77.44	0.96	0.96	0.96	25087	927	913
31	0.965	79.42	0.96	0.98	0.97	25364	1159	636
32	0.985	82.74	0.99	0.98	0.98	25478	345	522
33	0.982	73.27	0.98	0.99	0.99	25747	437	253
34	0.983	82.46	0.99	0.99	0.99	25818	320	182
35	0.978	79.12	1.00	0.95	0.97	24753	106	1247
36	0.977	81.77	0.99	0.97	0.98	25335	384	665
37	0.979	73.72	0.98	0.98	0.98	25593	517	407
38	0.978	80.68	0.96	0.98	0.97	25586	1035	414
39	0.988	80.88	0.98	0.99	0.99	25831	409	169
40	0.984	80.99	0.98	0.99	0.98	25698	570	302
41	0.967	80.57	0.99	0.94	0.97	24523	184	1477
42	0.988	83.08	0.98	0.99	0.99	25787	472	213
43	0.985	83.82	0.99	0.99	0.99	25703	329	297
44	0.983	82.65	0.98	0.99	0.99	25792	414	208
45	0.983	80.92	0.99	0.99	0.99	25628	296	372
46	0.977	82.59	0.99	0.98	0.98	25496	296	504
47	0.984	83.25	0.99	0.99	0.99	25830	322	170
48	0.987	82.66	0.99	0.99	0.99	25818	350	182
49	0.984	83.12	0.99	0.99	0.99	25765	309	235
50	0.984	83.15	0.99	0.99	0.99	25757	328	243
51	0.983	83.10	0.99	0.99	0.99	25698	312	302
52	0.984	82.99	0.99	0.99	0.99	25772	339	228

Los pesos en la época 29k serán los pesos a ser usados al presentar el mayor mAP así como en conjunto el menor número de Falsos positivos (FP) y Falsos negativos (FN).

6.4.3 Resultados

Los resultados al aplicar la clasificación letras en las 16 mil imágenes de validación se presentan en la Tabla 6.3, donde:

- **class_id**: es el valor número de la clase usado durante el entrenamiento.
- **name**: es la etiqueta usada para mostrar la clase durante la validación.
- **ap**: es la métrica “average precision” de la clase.

Tabla 6.3 Resultados del modelo YOLO en la clasificación de letras.

class_id = 0, name = leta, ap = 99.37 %	class_id = 13, name = letn, ap = 100.00 %
class_id = 1, name = letb, ap = 100.00 %	class_id = 14, name = leto, ap = 99.95 %
class_id = 2, name = letc, ap = 100.00 %	class_id = 15, name = letp, ap = 100.00 %
class_id = 3, name = letd, ap = 100.00 %	class_id = 16, name = letq, ap = 100.00 %
class_id = 4, name = lete, ap = 100.00 %	class_id = 17, name = letr, ap = 100.00 %
class_id = 5, name = letf, ap = 100.00 %	class_id = 18, name = lets, ap = 100.00 %
class_id = 6, name = letg, ap = 100.00 %	class_id = 19, name = lett, ap = 99.96 %
class_id = 7, name = leth, ap = 98.12 %	class_id = 20, name = letu, ap = 100.00 %
class_id = 8, name = leti, ap = 99.55 %	class_id = 21, name = letv, ap = 100.00 %
class_id = 9, name = letj, ap = 99.69 %	class_id = 22, name = letw, ap = 100.00 %
class_id = 10, name = letk, ap = 100.00 %	class_id = 23, name = letx, ap = 100.00 %
class_id = 11, name = letl, ap = 99.91 %	class_id = 24, name = lety, ap = 99.96 %
class_id = 12, name = letm, ap = 100.00 %	class_id = 25, name = letz, ap = 100.00 %

Indicador 3 letras clasificadas correctamente

Indicador 3 = C / T

C: clasificaciones correctas
T: clasificaciones totales

Indicador 3 = 99.87%

6.5 Experimentación y resultados del aplicativo móvil

6.5.1 Observaciones

Para la extracción del valor del dato personal al probar en un celular Android 4.4 (Kitkat) con RAM 1GB, CPU 1.2GH se observó que:

- Al desarrollar un aplicativo para un entorno móvil se deben tomar en cuenta la limitada capacidad computacional en procesamiento, almacenamiento y capacidad de batería, mayor uso del CPU mayor consumo de batería.
- Se observó que el modelo Adaboost al ser un modelo de clasificación binario y que requiere pocas características (22.1KB en pesos de los 26 modelo binarios) es muy rápido, para clasificar 10 letras, con 100 muestras de cada letra se demora 3 segundos aproximadamente.
- Se observo que YOLO v3 tiny al ser una red neuronal pesada (34118 KB en pesos) es más lenta en clasificar letras, aproximadamente 0.5 segundos por letra.
- Así mismo para reducir el ruido tanto para el modelo de detección como clasificación se evaluarán en 100 capturas de pantalla consecutivas para Adaboost y 2 para YOLO.

6.5.3 Resultados

Los resultados del indicador 4 e indicador 5 se calcularán en base a los demás indicadores.

Indicador 4 Letras correctamente clasificadas y posicionadas en el dato de la persona.

Indicador 4 = LC / LT

LC: Letras correctos
LT: Letras totales

Una letra se considerará clasificada y posicionada correctamente si es segmentada correctamente y clasificada correctamente, por lo cual el indicador 4 seria:

Indicador 4 = Indicador 1 * indicador 2 * indicador 3
Indicador 4 Adaboost = 100% * 99.01% * 88.52% = 87.64%
Indicador 4 Yolo v3 tiny = 100% * 99.01% * 99.87% = 98.88%

Indicador 5 datos correctamente identificados.

Indicador 5 = DC / DT

DC: Letras correctos
DT: Letras totales

Según se observó en los 85 datos de muestra la cantidad promedio de letras por dato es de 8. Por lo tanto, se asumirá que el indicador 5 (dato correctamente identificado), estará dado por la correcta identificación de sus 8 letras.

Indicador 5 = (Indicador 4) ^ 8
Indicador 5 Adaboost = (87.64%) ^ 8 = 34.80%
Indicador 5 Yolo v3 tiny = (98.88%) ^ 8 = 91.38%

7 Conclusiones y recomendaciones

7.1 Conclusiones

1. Es posible detectar correctamente las palabras y letras de un dato personal del DNI electrónico usando técnicas de procesamiento de imagen con una precisión del 100% para la detección de palabras y 99% para la detección de letras.
2. Es posible clasificar correctamente las letras de cada dato personal del DNI electrónico usando modelos de Adaboost y sus características de Haar, en un porcentaje del 88.52% y con YOLO V3 tiny en un 99.87%.
3. Es posible detectar correctamente el valor de un dato personal combinando el modelo de detección de palabras y letras con el modelo de clasificación de letras, en un porcentaje del 34.80% con Adaboost y 91.38% YOLO, YOLO con mejor rendimiento que los procesos manuales (75%) (Integrate, 2019).
4. Las etapas de preprocesamiento ayudan a reducir las características y con ello también reducir la complejidad y reducir el tiempo de procesamiento tanto para el entrenamiento como para la clasificación.
5. Cuando se requiera rapidez en la detección y clasificación se podría usar Adaboost mientras que si requerimos mayor precisión podrías usar “YOLO v3 tiny”.

7.2 Recomendaciones de trabajos futuros

- Se puede extender el aplicativo móvil para detectar otros documentos y otros datos del documento de identidad, por ejemplo, puntos característicos del rostro de la persona.
- La clasificación de imágenes se podría realizar con técnicas de aprendizaje profundo en caso se contase con mayor volumen de datos para el entrenamiento y este se realice en un entorno no móvil.
- El entrenamiento con más fuentes de letras podría dar mayor robustez al modelo de detección de letras y mejorar su rendimiento.

8 Bibliografía

ABBYY. (2019, Octubre 20). *ABBYY Business Card Reader*. From https://www.abbyybcr.com/en/

ABBYY. (2019, Octubre 20). *ABBYY FineReader 12 Professional*. From https://www.abbyy.com/en-apac/finereader12/en

Amazon. (2019, Octubre 20). *Amazon Rekognition*. From https://aws.amazon.com/rekognition/

Amirgaliyev, B., Kuatov, K., & Baibatyr, Z. (2017). Application of Convolutional Neural Network for Optical Character Recognition Designed for Kazakhstan Identity Cards. *2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT)* (pp. 1-3). Moscow, Russia, Russia: IEEE.

Anjerome, R., Sybingco, E., Quiros, A., Uy, R., Vicerra, E., & Dadios, E. (2016). Fuzzy logic based vehicular plate character recognition system using image segmentation and scale-invariant feature transform. *IEEE Conference Publications*, p. 676 - 681.

Cook, R., & Torrance, K. (1982). A reflectance model for computer graphics. *ACM Transactions on Graphics*, 7-24.

Fang, X., Fu, X., & Xu, X. (2017). ID card identification system based on image recognition. *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)* (pp. 1488-1492). Siem Reap, Cambodia: IEEE.

Glassner, A. (1995). Principles of Digital Image Synthesis. *Morgan Kaufmann Publishers*.

González, A., Bergasa, L., Yebes, J., & Bronte, S. (2012). A character recognition method in natural scene images. *Pattern Recognition (ICPR)* (pp. 621-624). Tsukuba, Japón: IEEE.

Gonzalez, R., & Woods, R. (2008). *Digital Image Processing.* Pearson.

Google. (2019, Octubre 20). *Google Cloud Platform*. From https://cloud.google.com/vision/

IBM. (2019, Octubre 20). *Visual Recognition*. From https://www.ibm.com/watson/services/visual-recognition/

ICAR. (2019, Octubre 20). *ID mobile*. From https://www.icarvision.com/es/id_mobile

Integrate. (2019, Octubre 20). *Integrate Indices.* From Data Quality - B2B Tech Industry: http://discover.integrate.com/hs-fs/hub/419314/file-2405295672-pdf/Integrate_Indices+Guides/IntegrateIndices_DataQualityB2BTechIndustry_v1.1.pdf

Isaza, C., Vargas, J., Gaviria, C., & Hernández, L. (2012). Automatic OCR system for colombian DNIs. *Image, Signal Processing, and Artificial Vision (STSIVA), 2012 XVII Symposium of* (pp. 295-300). Antioquia: IEEE.

J, B., Chen, Z., Feng, B., & Xu, B. (2014). Image character recognition using deep convolutional neural network learned from different languages. *Image Processing (ICIP)* (pp. 2560-2564). Paris, Francia: IEEE.

Liem, H. D., Minh, N. D., Trung, N. B., Duc, H. T., Hiep, P. H., Dung, D. V., & Vu, D. H. (2019). FVI: An End-to-end Vietnamese Identification Card Detection and Recognition in Images. *2018 5th NAFOSTED Conference on Information and Computer Science (NICS)* (pp. 338-340). Ho Chi Minh City, Vietnam: IEEE.

López, A., Valveny, E., & M, V. (2019, Octubre 20). *Detección de Objetos por la Univeridad Autonoma de Barcelona*. From Coursera: https://www.coursera.org/learn/deteccion-objetos

Microsoft. (2019, Octubre 20). *Computer Vision*. From https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/

Naiemi, F., Ghods, V., & Khalesi, H. (2019). An efficient character recognition method using enhanced HOG for spam image detection. *Soft Computing*, 1-16.

Nuance. (2019, Octubre 20). *OmniPage Ultimate*. From http://www.nuance.es/empresas/producto/omnipage/ultimate/index.htm

Nuance. (2019, Octubre 20). *Paperport Professional 14*. From http://www.nuance.es/empresas/producto/paperport/professional/index.htm

Pascal. (2019, Octubre 20). *The PASCAL VOC project*. From http://host.robots.ox.ac.uk/pascal/VOC/

Pratama, M., Satyawan, W., Fajar, B., Fikri, R., & Hamzah, H. (2018). Indonesian ID Card Recognition using Convolutional Neural Networks. *2018 5th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)* (pp. 178-181). Malang, Indonesia, Indonesia: IEEE.

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 7263-7271). Honolulu: IEEE.

Redmon, J., & Farhadi, A. (2019, Octubre 20). *Darknet: Open Source Neural Networks in C.* From https://pjreddie.com/darknet/

Redmon, J., & Farhadi, A. (2019, Octubre 20). *YOLO: Real-Time Object Detection*. From https://pjreddie.com/darknet/yolo/

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 779-788). Las Vegas: IEEE.

RENIEC. (2019, Octubre 20). *DNI Electrónico*. From https://portales.reniec.gob.pe/web/dni

Sinhal, K. (2019, Octubre 20). *Object Detection using Deep Learning for advanced users (Part-1).* From Linkedin: https://www.linkedin.com/pulse/object-detection-using-deep-learning-advanced-users-part-1-sinhal/

Software, L. (2019, Octubre 20). *SODA PDF*. From https://www.sodapdf.com

Su, B., Lu, S., Phan, T., & Tan, C. (2012). Character extraction in web image for text recognition. *Pattern Recognition (ICPR), 2012* (pp. 3042-3045). Tsukuba, Japón: IEEE.

Torreance, K., & Sparrow, E. (1967). Theory for off-specular reflection from roughened surfaces. *Journal of the Optical Society of America*, 1105-1114.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, 2001. CVPR 2001* (pp. I-511-I-518 vol.1). Kauai, HI, USA, USA : IEEE.

Xiang, H.-R., Peng, J., Ma, Y., He, Y., Zheng, Z.-Z., Mou, F., & Li, J. (2018). Chinese Character Recognition Based on Residual Separable Convolutional Neural Network. *2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)* (pp. 49-53). Chengdu, China, China: IEEE.

Xu, J., & Wu, X. (2018). A System to Localize and Recognize Texts in Oriented ID Card Images. *2018 IEEE International Conference on Progress in Informatics and Computing (PIC)* (pp. 149-153). Suzhou, China, China: IEEE.

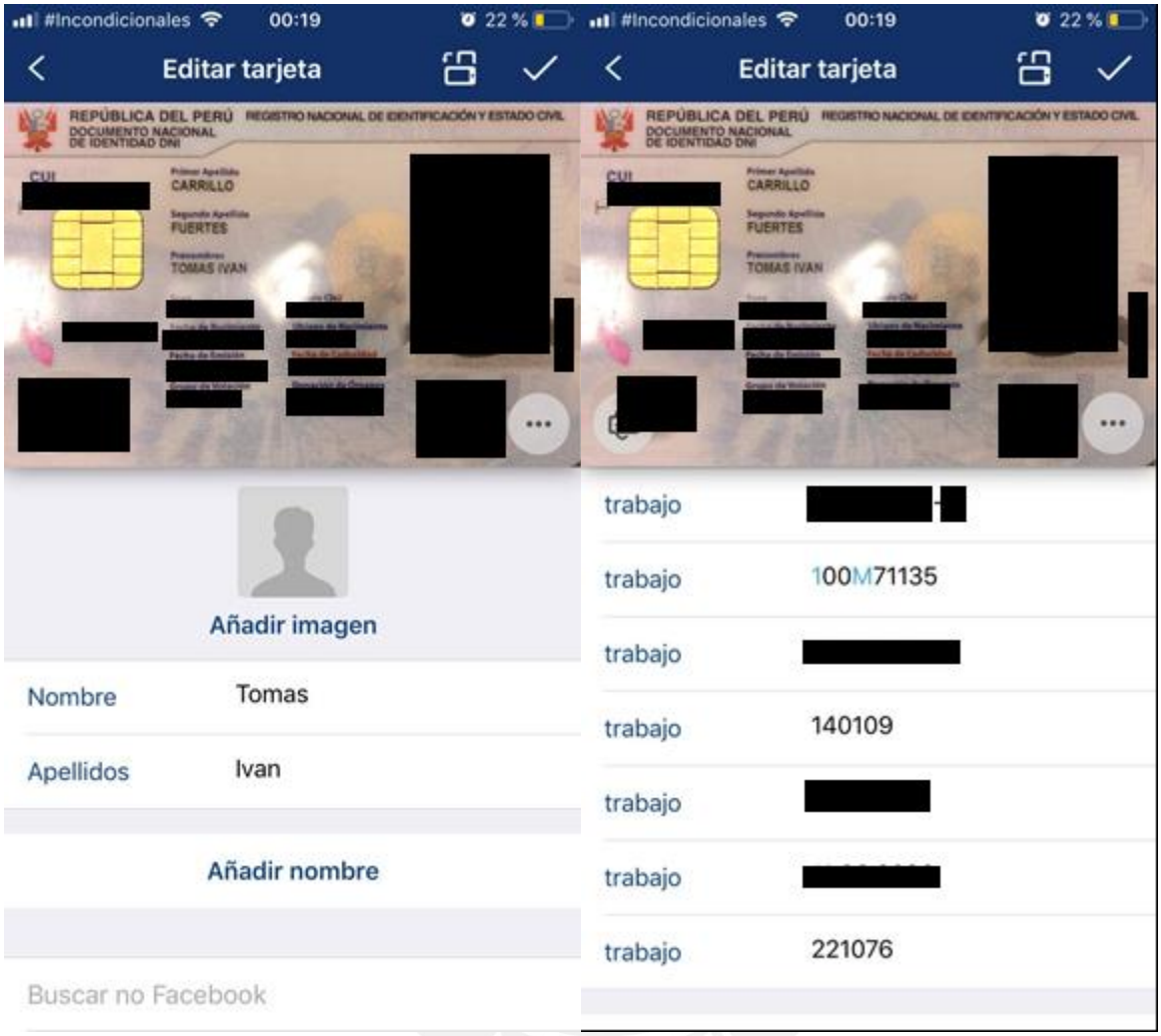
Anexos.

Actividades:

1. Etiquetado de DNIs (terminado)
2. Generación de data sintética (terminado)
3. Extracción de características con Haar (terminado)
4. Clasificación con AdaBoost (terminado para letra A y R) 2/26, 8%
5. Desarrollo de aplicativo
6. Validación de resultados
7. Escritura de tesis/artículo

Activ.	1 21-08	2 28-08	3 04-09	4 11-09	5 18-09	6 25-09	7 02-10	8 09-10	9 16-10	10 23-10	11 30-10	12 06-11	13 13-11	14 20-11
1	x	x	x											
2		x	x											
3				x	x	x								
4					x	x	x	x						
5						x	x	x	x	x	x	x		
6									x	x	x	x		
7						x	x	x	x	x	x	x	x	x

ABBY OCR en DNI electrónico peruano:



Detalle de la configuración “yolov3-tiny-obj.cfg” para entrar YOLO

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=4
width=96
height=96
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 52000
policy=steps
steps=41600,46800
scales=.1,.1

[convolutional]
batch_normalize=1
filters=16
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2
```

```
[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=1

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

###########

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky
```

```
[convolutional]
size=1
stride=1
pad=1
filters=93
activation=linear

[yolo]
mask = 3,4,5
anchors = 10,14,  23,27,  37,58,
81,82,  135,169,  344,319
classes=26
num=6
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1

[route]
layers = -4

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[upsample]
stride=2

[route]
layers = -1, 8

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=93
activation=linear

[yolo]
mask = 0,1,2
anchors = 10,14,  23,27,  37,58,
81,82,  135,169,  344,319
classes=26
num=6
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
```